CHƯƠNG 4

Styles, Themes, và CSS

Thiết kế giao diện người dùng thách thức chúng tôi để bố trí thanh lịch và chức năng chức năng và điều khiển trong khi duy trì một thiết kế mạch lạc và một trải nghiệm người dùng nhất quán trên toàn bộ ứng dụng. Chương này không phải là về việc tạo ra các ứng dụng đẹp và trực quan hấp dẫn, mà là một nghệ thuật trong và của chính nó. Chương này khám phá cách đẹp và hấp dẫn để kiến trúc sư Code giao diện người dùng của bạn, tách bản trình bày từ nội dung và cấu trúc bằng cách sử dụng Resources, Style, Themes, và Cascading Style Sheets. Tạo ra thiết kế đẹp là một cuộc hành trình, và mục đích của chương này là để đưa bạn vào một cuộc hành trình từ một trang đơn giản nhưng chức năng để một trong đó cũng là đẹp, Dynamic, và duy trì. Reusability của UI Code củafers nhất quán của thiết kế, sạch hơn và dễ đọc hơn Code, và sự nhanh nhẹn để thay đổi giao diện của các ứng dụng theo thời gian.

Chúng ta sẽ bao gồm toàn bộ câu chuyện về định dạng giao diện người dùng Xamarin: bắt đầu với Style thủ công, Resources và Dictionaries, Style, cácMES, và sau đó Cascading Style Sheets (CSS). Trước tiên, chúng ta sẽ tạo một trang mà không có Style và sau đó Style bằng tay. Sau đó, thay vì tự hardcoding các thiết lập bất động sản, chúng tôi sẽ sử dụng các Resources được lưu trữ tập trung trong một Dictionaries và tìm chúng lên statically và động. Chúng ta sẽ kiểm tra các Style và cách sử dụng chúng Explicit trên một View hoặc ngầm trên tất cả các trường hợp của một View. Chúng tôi sẽ khám phá Style thừa kế và kỹ thuật ghi đè. Chúng tôi sẽ View xét các Themes Xamarin. Forms Light và Dark và làm thế nào để sử dụng chúng như là cơ sở của một Tùy chỉnh Themes và, cuối cùng, sử dụng CSS như là một thay thế cho XAML để Style một trang.

Ứng dụng demo của chúng tôi là một trang phản hồi cho phép người dùng chia sẻ trải nghiệm của họ bằng cách sử dụng ứng dụng.

**Tạo trang không có Style**

Chúng tôi sẽ bắt đầu với một trang cơ bản mà không có Style cho phép người dùng cung cấp phản hồi. Nó sẽ có các trường Themes và Submit và Button gửi và Cancel. Chẳng bao lâu chúng tôi sẽ Style trang.

Tạo một mẫu XAML trang được gọi là feedbackpage. XAML với một mục nhập đại diện cho dòng Themes và View biên tập cho các phản hồi. Cuối cùng, thêm một Button Submit và Cancel vào form để xử lý các tương tác của người dùng. Để cấu trúc nội dung, sử dụng bố cục Grid với một cột và ba hàng, một cho dòng Themes, một cho phản hồi và cuối cùng cho một stacklayout theo chiều dọc places hai Button, như được hiển thị trong Liệt kê 4-1.

***Liệt kê 4-1.*** Trang phản hồi đơn giản

<ContentPage Title="Plain" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.FeedbackPage">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Entry Placeholder="Enter Subject" Grid.Row="0" />

<Editor x:Name="Feedback" Grid.Row="1" Unfocused="HandleFeedback" Focused="HandleFeedback" />

<StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Cancel" />

<Button Text="Submit" />

</StackLayout>

</Grid>

</ContentPage>

Grid và stacklayout cung cấp cấu trúc cho trang và nội dung bao gồm mục nhập, trình chỉnh sửa và lượt View Button. Stacklayout được sử dụng với một hướng dọc để đảm bảo rằng hai Button được căn chỉnh bên dưới nhau. Mục, trình chỉnh sửa và stacklayout được định vị trong Grid sử dụng thuộc tính Grid. Row đính kèm. Grid. Column là không cần thiết ở đây, cho rằng Grid chỉ có một cột. Cột Grid sử dụng sao để tiêu thụ tất cả không gian có sẵn, có nghĩa là, ColumnDefinition width = "\*".

Các hàng Grid đầu tiên và thứ ba chỉ sử dụng nhiều không gian cần thiết cho dòng Themes và các Button gửi và hủy, trong khi phần còn lại của không gian dọc được phân bổ cho khu vực phản hồi bằng cách sử dụng và rowdefinition height = "\*".

View biên tập không có thuộc tính Placeholder như View mục . Tạo văn bản Placeholder bằng tay bằng cách gán một phương pháp handlefeedback cho cả thuộc tính focuSED và không tập Trung . Tạo trình xử lý sự kiện handlefeedback trong Code sau đó đặt văn bản của trình chỉnh sửa khi nó tăng hoặc mất tập trung, như được hiển thị trong danh sách 4-2.

***Liệt kê 4-2.*** Trang phản hồi đơn giản

public partial class FeedbackPage : ContentPage {

const string placeHolderText = "Type your message here";

void HandleFeedback(object sender, Xamarin.Forms.FocusEventArgs e) { var text = Feedback.Text;

if (Feedback.Text Feedback.Text return;

}

if (Feedback.Text Feedback.Text return;

}

}

public FeedbackPage()

== placeHolderText) { = string.Empty;

== string.Empty) {

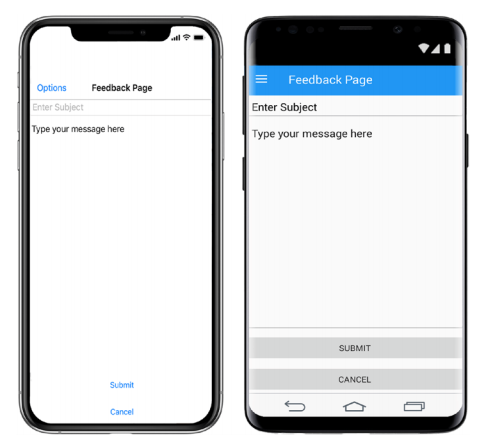
= placeHolderText;

InitializeComponent();

Feedback.Text = placeHolderText; }

}

Cho mỗi phần tử XAML được đặt tên, phân tích cú pháp XAML tạo ra một biến cục bộ trong tệp kết quả tạo ra. Khi trang được khởi tạo, thuộc tính văn bản của trình chỉnh sửa , được gọi là phản hồi trong XAML, được đặt thành một placeholdertext. Khi trình biên tập gains hoặc mất tập trung, phương pháp Handlefeedback được gọi là kiểm tra View văn bản biên tập vẫn là placeholdertext và đổ nó cho người dùng để nhập thông tin phản hồi. Nếu trình biên tập mất tập trung và vẫn còn trống, nó đặt văn bản trở lại placeholdertext. Điều này bắt chước các chức năng tích hợp của chế độ View mục , trong đó có này ra khỏi hộp. Trong dự án App.cs, bọc các feedbackpage bên trong một navigationpage và thêm nó vào mainpage của ứng dụng. Hình 4-1 cho thấy kết quả cho cả hai nền tảng.



***Hình 4-1.*** *Trang phản hồi đơn giản*

Trang phản hồi có chức năng và Color xám. Hãy thêm một số Color.

Tạo kiểu thủ công bằng cách sử dụng tính định dạng View

Cải thiện sự xuất hiện của Feedbackpage bằng cách cung cấp Color cho văn bản và nguồn gốc, sử dụng bố cục và View các thuộc tính định dạng.

1. Tạo một trang XAML mới được gọi là Localdesignpage. XAML và COPy XAML qua từ feedbackpage. XAML.

2. Sử dụng thuộc tính Backgroundcolor của contentpage và Grid để tách riêng các biểu mẫu Canvas đại diện bởi Grid từ trang nền bằng cách làm cho Grid Color xám và nền contentpage xanh.

< ContentPage BackgroundColor = "#4CAF50" >

< Grid BackgroundColor = "#AAAAAA" >

1. Sử dụng Grid padding và margin thuộc tính để cung cấp một số không gian cho nội dung cách xa bezel màn hình.

< Grid padding = "30" margin = "30" >

1. Bao quanh lĩnh vực đầu vào phản hồi với một Stacklayout có một nền Color bạc và chỉ là một pixel lớn hơn điều khiển biên tập viên .

<StackLayout Orientation="Vertical" Padding="1" BackgroundColor="Silver">

1. Cung cấp cho các lĩnh vực phản hồi đầu vào tất cả các SpaCE nó có thể nhận được theo chiều dọc.

< Editor VerticalOptions = "FillAndExpand" >

6. Thêm TextColor và fontattributes vào dòng Themes và nhập thông tin phản hồi trường.

<Entry TextColor="Navy" FontAttributes="Italic" /> <Editor TextColor="Navy" FontAttributes="Italic" />

7. Style các Button để giúp người dùng đưa ra quyết định đúng bằng cách làm cho Button gửi Color xanh lá cây và Button hủy bỏ Color cam với văn bản đậm sử dụng tất cả các không gian họ có thể nhận được theo chiều ngang, như trong danh sách 4-3.

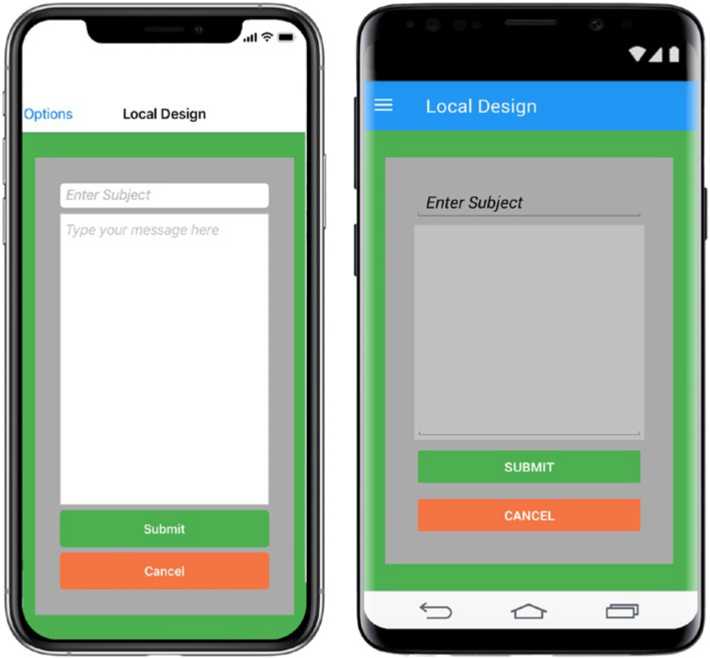
***Liệt kê 4-3.*** Trang phản hồi đơn giản

<Button Text="Submit" TextColor="White" FontAttributes="Bold" BackgroundColor="#4CAF50" />

<Button Text="Cancel" HorizontalOptions="FillAndExpand" TextColor="White"

FontAttributes="Bold" BackgroundColor="#f47442" />

Bây giờ chúng ta có một trang phản hồi đầy Colors, như được hiển thị trong hình 4-2 cho cả hai nền tảng.



***Hình 4-2.*** *Trang phản hồi theo kiểu*

Code đầy đủ: trang phản hồi sử dụng định dạng View

Thuộc tính

Danh sách 4-4 cung cấp Code hoàn chỉnh cho localdesignpage. XAML.

***Liệt kê 4-4.*** Trang phản hồi theo Styled

<ContentPage BackgroundColor="#4CAF50" Title="Local Design" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.LocalDesignPage">

<Grid Padding="30" Margin="30" BackgroundColor="#AAAAAA"> <Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" /> </Grid.ColumnDefinitions> <Entry Placeholder="Enter Subject" Grid.Row="0" TextColor="Navy" FontAttributes="Italic" />

<StackLayout Orientation="Vertical" Grid.Row="1" Padding="1" BackgroundColor="Silver">

<Editor x:Name="Feedback" VerticalOptions="FillAndExpand" TextColor="Navy" FontAttributes="Italic" Unfocused="HandleFeedback"

Focused="HandleFeedback" /> </StackLayout>

<StackLayout Orientation="Vertical" Grid.Row="2"> <Button Text="Submit" HorizontalOptions="FillAndExpand" TextColor="White" FontAttributes="Bold" BackgroundColor="#4CAF50" />

<Button Text="Cancel" HorizontalOptions="FillAndExpand"

TextColor="White" FontAttributes="Bold" BackgroundColor="#f47442" />

</StackLayout>

</Grid>

</ContentPage>

Số dòng vẫn gần như tương tự nhưng XAML bây giờ lộn xộn với Values hardcoded. So với điều này, trang unstyled trong Listing 4-1 là ngắn.

Resources có thể giải quyết vấn đề hardcoding. Hãy View như thế nào.

**Resources và Dictionaries**

*Resources* là một đối tượng trong Dictionaries có thể được lấy bởi khóa chuỗi của nó. Bất kỳ yếu tố XAML, thực tế bất kỳ C# lớp, có thể được định nghĩa là Resources. Resources phổ biến XAML là mẫu, kiểu, Values thường sử dụng và chuyển đổi Values.

*Resources Dictionaries* là một kho lưu trữ Values quan trọng. Các phím là chuỗi và các Values bất kỳ loại đối tượng. Resources Dictionaries đóng vai trò quan trọng trong hệ thống tạo kiểu của Xamarin. Forms. Họ cho phép tách các bài trình bày từ nội dung và cấu trúc của một trang.

Tra cứu Resources trong Dictionaries có thể Static hoặc động. Tìm kiếm Static chỉ xảy ra một lần khi View được tạo ra. Các thay đổi Resources không được áp dụng cho việc View. Tra cứu động cho phép Resources để thay đổi tại thời gian chạy và thay đổi có thể được phản ánh trong dạng View.

Resources Dictionaries có thể được định nghĩa ở các cấp ứng dụng, trang và View. Resources được xác định tại địa phương có thể ghi đè Resources được xác định ở cấp trang hoặc ứng dụng. Tiêu chuẩn hướng dẫn phạm vi áp dụng ở đây: thử và sử dụng Dictionaries View hoặc Resources cấp trang trước khi xác định chúng ở cấp độ ứng dụng để tạo ra Globals.

Resources Dictionaries Mẹo có thể được lưu trữ dưới dạng tệp XAML riêng biệt, có thể kết hợp.

Trang, bố cục và điều khiển, tất cả dạng View trong Xamarin. Forms phân lớp Visualelement, có thể chứa các định nghĩa Resources. Visualelement có một thành viên được gọi là nguồn Resources của loại resourcedictionary cho mục đích này.

**Xác định Resources**

Resources là các đối tượng được lưu trữ trong Resources Dictionaries có thể được lấy bằng phím của chúng. Xác định Resources bằng cách thêm Dictionaries nguồn lực vào trang, bố cục hoặc kiểm soát và thêm Resources vào Dictionaries đó.

Ví dụ, sử dụng các yếu tố thuộc tính cú pháp Contentpage. Resources để deFine một resourcedictionary với các Resources có thể truy cập vào trang feedbackpage và thêm một Color Resources đại diện cho trang backgroundcolor, như được hiển thị trong danh sách 4-5.

***Liệt kê 4-5.*** Xác định Resources bên trong Resources Dictionaries

<ContentPage.Resources> <ResourceDictionary>

<Color x:Key="PageBgColor">#4CAF50</Color> </ResourceDictionary>

</ContentPage.Resources>

Color với pagebgcolor chính bây giờ có thể được tham chiếu trong trang.

Mẹo tên Resources dựa trên những gì họ được sử dụng cho và không phải những gì họ đang có, ví dụ như, nukeButton thay vì redButton. điều này đảm bảo rằng chúng được sử dụng cho những gì họ đã dành cho.

Bạn có thể sử dụng các tên biến từ khoá khác, chẳng hạn như tên từ khóa XAML hoặc bất kỳ thứ gì bạn chọn, thay vì khóa, ví dụ:

< Color x:Name = "PageBgColor" > #4CAF50 </Color >

Điều này sẽ tạo Pagebgcolor biến địa phương trong C# Code phía sau. Đó là khuyến cáo, Tuy nhiên, để sử dụng các phím và để truy cập vào các nguồn lực trong Code đằng sau, h một Dictionaries tìm kiếm đi qua trong phím và sau đó đúc các đối tượng kết quả với loại mục tiêu, ví dụ:

var pageBgColor = (Color) Resources ["PageBgColor"];

Tạo Resources cho tất cả các Colors, Font Styles và thậm chí literals chuỗi được sử dụng làm trình giữ khách, như hiển thị trong danh sách 4-6.

***Liệt kê 4-6.*** Xác định Resources bên trong Resources Dictionaries

<ResourceDictionary>

<x:String x:Key="SubjectPlaceholder">Enter Subject</x:String> <x:String x:Key="FeedbackPlaceholder">Enter your message here </x:String>

<x:String x:Key="InputPlaceholder">Enter Subject</x:String> <Color x:Key="PageBgColor">#4CAF50</Color>

<Color x:Key="FormBgColor">#AAAAAA</Color>

<Color x:Key="InputTextColor">Navy</Color>

<Color x:Key="InputPlaceholderColor">Silver</Color>

<Color x:Key="FeedbackBorderColor">Silver</Color> <FontAttributes x:Key="InputFontStyle">Italic</FontAttributes> <Color x:Key="BtTextColor">White</Color>

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color> <FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> </ResourceDictionary>

Resources Dictionaries này là nội tuyến với Contentpage. Resources. Ngay sau đó chúng tôi sẽ phá vỡ nó ra vào tập tin XAML riêng của mình để nó có thể được sử dụng bởi các trang khác nhau.

Sau khi xác định Resources của bạn, bạn cần tham khảo chúng. Chúng tôi sẽ bắt đầu bằng cách sử dụng phần mở rộng đánh dấu Xamarin. Forms staticresource .

**Tra cứu Resources Static**

Phần mở rộng đánh dấu Staticresource gán Values cho một thuộc tính tài sản Bindable của một yếu tố XAML một lần, khi đối tượng XAML được tạo ra tại thời gian chạy. Bất kỳ thay đổi tiếp theo Resources không được phản ánh trên các đối tượng, và để tìm kiếm là Static. Sử dụng tra cứu Static nếu Resources không thay đổi trong thời gian cuộc sốngcủa ứng dụng. Điều này cải thiện hiệu suất của ứng dụng. Sử dụng tìm kiếm động thay vì nếu Resources được gán cho dạng View nhằm thay đổi, ví dụ: để thay đổi kiểu của dạng View tùy thuộc vào tương tác của người dùng.

Lưu ý StaticResources ném một ngoại lệ tại thời gian chạy nếu khoá không tồn tại.

Sử dụng phần mở rộng đánh dấu Staticresource để đặt Color nền của trang với tra cứu của pagebgcolor chính trong Resources Dictionaries. Sử dụng thuộc tính chính của staticresource PaSS trong khóa Resources cần được sử dụng, ví dụ:

<ContentPage BackgroundColor="{StaticResource Key=PageBgColor}" ...>

Bạn đã học trong chương 2 rằng một tài sản có thể được bỏ qua nếu tuyên bố là một tài sản nội dung. Do đó, việc kê khai trước là giống như

<ContentPage BackgroundColor="{StaticResource PageBgColor}" ...>

Tạo một trang XAML mới và gọi nó Staticresourcespage. XAML. Nhân đôi trang localdesignpage. XAML để bắt đầu với. Thay thế tất cả các thuộc tính trong XAML với tìm kiếm khoá tương ứng mà bạn đã xác định trong Resources Dictionaries trong danh sách 4-6 sử dụng phần mở rộng đánh dấu staticresource , như được hiển thị trong danh sách 4-7. Định nghĩa cột Grid và hàng được bỏ qua cho ngắn gọn.

***Liệt kê 4-7.*** Sử dụng StaticResource trong XAML thay vì hardcoded Values

<Grid Padding="30" Margin="30" BackgroundColor="{StaticResource FormBgColor}">

<Entry Placeholder="{StaticResource SubjectPlaceholder}" Grid.Row="0"

TextColor="{StaticResource InputTextColor}" FontAttributes=" {StaticResource

InputFontStyle}" /> <StackLayout Orientation="Vertical" Grid.Row="1" Padding="1" BackgroundColor="{StaticResource FeedbackBorderColor}">

<Editor x:Name="Feedback" VerticalOptions="FillAndExpand" TextColor="{StaticResource InputTextColor}" FontAttributes=" {StaticResource

InputFontStyle}" Unfocused="HandleFeedback" Focused="HandleFeedback"

/>

</StackLayout> <StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Submit" HorizontalOptions="FillAndExpand"

TextColor="{StaticResource BtTextColor}" FontAttributes=

"{StaticResource

BtFontStyle}" BackgroundColor="{StaticResource BtSubmitColor}" /> <Button Text="Cancel" HorizontalOptions="FillAndExpand" TextColor="{StaticResource BtTextColor}" FontAttributes=" {StaticResource

BtFontStyle}" BackgroundColor="{StaticResource BtCancelColor}" /> </StackLayout>

</Grid>

Thay thế Code sau StaticResourcesPage.XAML.CS Values hardcoded cho trình Placeholder phản hồi và Color văn bản phản hồi với tìm kiếm các Resources feedback giữ và inputtextcolor, như được hiển thị trong danh sách 4-8.

***Liệt kê 4-8.*** Sử dụng StaticResource bên trong Code phía sau thay vì hardcoded Values

public partial class StaticResourcesPage : ContentPage {

void HandleFeedback(object sender, Xamarin.Forms.FocusEventArgs e){ var placeHolderText = (string)Resources["FeedbackPlaceholder"];

if (Feedback.Text == placeHolderText) {

Feedback.Text = string.Empty;

Feedback.TextColor = (Color)Resources["InputTextColor"]; return;

}

if (Feedback.Text == string.Empty) {

Feedback.Text = placeHolderText;

Feedback.TextColor = (Color)Resources["InputPlaceholderColor"]; return;

}

}

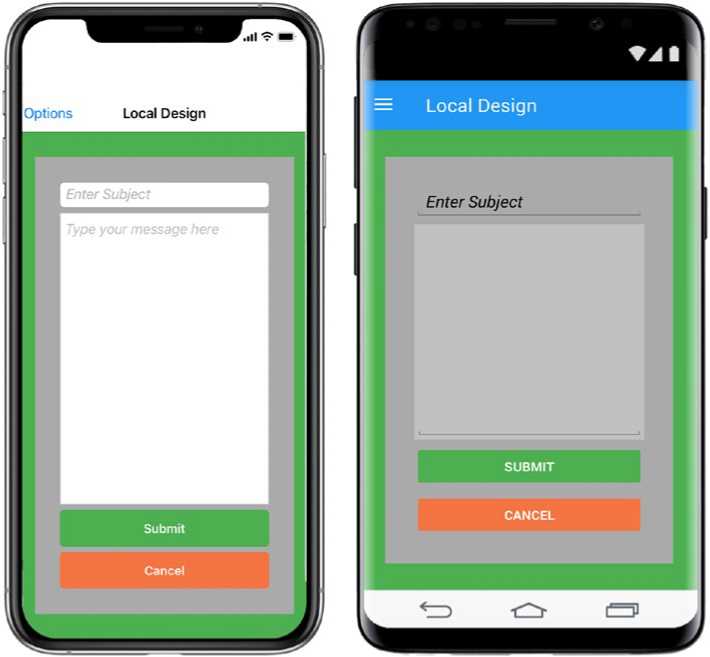
public StaticResourcesPage() {

InitializeComponent();

Feedback.Text = (string)Resources["FeedbackPlaceholder"];

Feedback.TextColor = (Color)Resources["InputPlaceholderColor"]; }

Trang phản hồi ở Figure 4-3 trông giống hệt như hình 4-2 với Style hardcoded. Cải tiến là trong kỹ thuật Code hóa.



***Hình 4-3.*** *Trang phản hồi theo kiểu sử dụng Resources và tra cứu Static*

Code đầy đủ: trang phản hồi sử dụng Resources Static

Danh sách 4-9 và danh sách 4-10 cung cấp Code hoàn chỉnh cho staticresourcespage. XAML và StaticResourcesPage.XAML.CS.

***Liệt kê 4-9.*** Staticresourcespage. XAML trong dự án mẫu

<?xml version="1.0" encoding="UTF-8"?>

<ContentPage BackgroundColor="{StaticResource Key=PageBgColor}"

Title="StaticResource" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.StaticResourcesPage">

<ContentPage.Resources>

<ResourceDictionary>

<x:String x:Key="SubjectPlaceholder">Enter Subject</x:String> <x:String x:Key="FeedbackPlaceholder">Type your message here</ x:String>

<x:String x:Key="InputPlaceholder">Enter Subject</x:String>

<Color x:Key="PageBgColor">#4CAF50</Color>

<Color x:Key="FormBgColor">#AAAAAA</Color>

<Color x:Key="InputTextColor">Navy</Color>

<Color x:Key="InputPlaceholderColor">Silver</Color>

<Color x:Key="FeedbackBorderColor">Silver</Color> <FontAttributes x:Key="InputFontStyle">Italic</FontAttributes> <Color x:Key="BtTextColor">White</Color>

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color> <FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> </ResourceDictionary>

</ContentPage.Resources>

<Grid Padding="30" Margin="30" BackgroundColor="{StaticResource FormBgColor}"> <Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*" />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Entry Placeholder="{StaticResource SubjectPlaceholder}" Grid.Row="0" TextColor="{StaticResource InputTextColor}" FontAttributes="{StaticResource InputFontStyle}" /> <StackLayout Orientation="Vertical" Grid.Row="1" Padding="1" BackgroundColor="{StaticResource FeedbackBorderColor}"> <Editor x:Name="Feedback" VerticalOptions="FillAndExpand" TextColor="{StaticResource InputTextColor}"

FontAttributes="{StaticResource InputFontStyle}"

Unfocused="HandleFeedback" Focused="HandleFeedback" /> </StackLayout>

<StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Submit" HorizontalOptions="FillAndExpand" TextColor="{StaticResource BtTextColor}" FontAttributes="{StaticResource BtFontStyle}" BackgroundColor="{StaticResource BtSubmitColor}" /> <Button Text="Cancel" HorizontalOptions="FillAndExpand" TextColor="{StaticResource BtTextColor}" FontAttributes="{StaticResource BtFontStyle}" BackgroundColor="{StaticResource BtCancelColor}" /> </StackLayout>

</Grid>

</ContentPage>

***Liệt kê 4-10.*** StaticResourcesPage.XAML.CS trong dự án Forms

using Xamarin.Forms;

namespace StyleExamples {

public partial class StaticResourcesPage : ContentPage {

void HandleFeedback(object sender, Xamarin.Forms.FocusEventArgs e) { var placeHolderText = (string)Resources["FeedbackPlaceholder"];

if (Feedback.Text == placeHolderText) {

Feedback.Text = string.Empty;

Feedback.TextColor = (Color)Resources["InputTextColor"]; return;

}

if (Feedback.Text == string.Empty) {

Feedback.Text = placeHolderText;

Feedback.TextColor = (Color)Resources["InputPlaceholder Color"];

return;

}

}

public StaticResourcesPage() {

InitializeComponent();

Feedback.Text = (string)Resources["FeedbackPlaceholder"]; Feedback.TextColor = (Color)Resources["InputPlaceholderColor"];

}

}

}

Resources Static cải thiện đáng kể khả năng bảo trì của Code vì không có gì là hardcoded, Code được tái sử dụng, và Style được tách ra từ giao diện người dùng. Tuy nhiên, XAML ít có thể đọc được so với hai lặp trước đó mà chúng tôi đã không Styling hoặc chỉ hardcoded Style. Quan trọng nhất, việc tra cứu Resources được thực hiện một lần bởi staticresourcevà bất kỳ thay đổi nào trong Resources không được áp dụng cho nội dung trang.

Hãy khắc phục sự cố sử dụng phần mở rộng đánh dấu dynamicresource .

**Dynamic Resource tra cứu**

Phần mở rộng đánh dấu Dynamicresource gán Values cho một thuộc tính tài sản Bindable của một yếu tố XAML. Thay đổi cho Resources có thể được áp dụng cho thuộc tính lúc chạy; do đó, việc tra cứu được gọi là động. Sử dụng tra cứu động để đảm bảo rằng một View được Cập Nhật nếu Resources của nó thay đổi để đáp ứng với một tương tác người dùng. Tra cứu động là nguồn lực chuyên sâu và có thể ảnh hưởng đến hiệu suất. Thực hành tốt nhất là sử dụng tra cứu Static cho đến khi cần phải thay đổi một nguồn lực tại thời gian chạy.

Lưu ý một DynamicresourCE không cần phải tồn tại khi tuyên bố XAML. ứng dụng có thể thêm khoá và đối tượng tại thời gian chạy. XAML sử dụng Values mặc định của thuộc tính tài sản Bindable nếu không có Resources có thể tìm thấy.

Như với Staticresource bạn có thể bỏ qua các tài sản chính của dynamicresource. Sử dụng phần mở rộng đánh dấu dynamicresource để thay thế Color nền của trang bằng tra cứu của pagebgcolor chính trong Resources Dictionaries, ví dụ:

< ContentPage BackgroundColor = "(DynamicResource PageBgColor}"... >

Tạo một trang XAML mới và gọi trang mới dynamicresourcepage. XAML. Sao chép Code từ staticresourcespage. XAML vào trang mới của bạn. Thay thế tất cả các phần mở rộng đánh dấu staticresource bằng dynamicresource.

Button "hủy" và "gửi" của chúng tôi đầy Colors và có thể nhấp. Điều gì nếu chúng tôi hướng dẫn sự tương tác hơn một chút bằng cách ghép chúng cho đến khi người dùng ít nhất nhập vào một Themes? Thêm một vài Resources Color khác vào Resources Dictionaries cho các Color nền khác nhau cho trang và Button tùy thuộc vào việc người dùngcó phải là Themes hay không và khởi tạo các Button khi không hoạt động, như được hiển thị trong Liệt kê 4-11.

***Liệt kê 4-11.*** Resources để xử lý tương tác người dùng

<Color x:Key="PageBgColorNoSubject">#85af87</Color>

<Color x:Key="PageBgColorWithSubject">#4CAF50</Color>

<Color x:Key="BtSubmitActiveColor">#4CAF50</Color>

<Color x:Key="BtCancelActiveColor">#f47442</Color>

<Color x:Key="BtInactiveColor">Gray</Color>

Tiếp theo, gán pagebgcolor, btsubmitcolor và btcancelcolor Dynamically tùy thuộc vào việc có một dòng Themes hay không.

Thay đổi một nguồn lực được thực hiện trong Code phía sau, như thế này:

Resources ["PageBgColor"] = Resources ["PageBgColorNoSubject"];

Thêm trình xử lý sự kiện Handlesubject để View mục để đáp ứng để đạt được và mất tập trung để thay đổi Color Resources dựa trên tương tác của người dùng, ví dụ:

<Entry Unfocused="HandleSubject" Focused="HandleSubject" />

Cả hai sự kiện không tập trung và tập trung có dây ngay bây giờ để xử lý handlesubject . Tạo phương thức handlesubject và kiểm tra View dòng Themes có bất kỳ văn bản nào hay không. Nếu không có văn bản được cung cấp, thay đổi pagebgcolor thành pagebgcolornosubject và btsubmitcolor và btcancelcolor để btinactivecolhoặc và quay lại PageBgColorWithSubject và btsubmitactivecolor và btcancelactivecolor ngay sau khi người dùng đã thay đổi văn bản và mục nhập View mất tập trung, như được hiển thị trong danh sách 4-12.

***Liệt kê 4-12.*** Thay đổi đối tượng Resources dựa trên tương tác người dùng

void HandleSubject(object sender, Xamarin.Forms.FocusEventArgs e) {

if (Subject.Text == string.Empty) {

Resources["PageBgColor"] = Resources["PageBgColorNoSubject"];

Resources["BtSubmitColor"] = Resources["BtInactiveColor"];

} else {

Resources["PageBgColor"] = Resources["PageBgColorWithSubject"]; Resources["BtSubmitColor"] = Resources["BtSubmitActiveColor"];

}

}

Cuối cùng, khởi tạo constructor trang để đảm bảo rằng biểu mẫu ở trạng thái không hoạt động khi nó được truy cập lần đầu tiên, như được hiển thị trong danh sách 4-13.

***Liệt kê 4-13.*** Dynamicresourcespage Constructor

public DynamicResourcesPage() {

InitializeComponent();

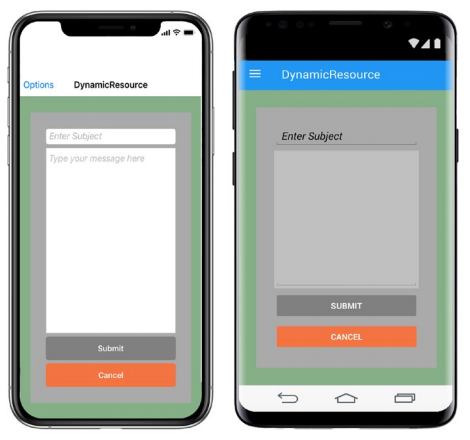
Feedback.Text = (string)Resources["FeedbackPlaceholder"];

Feedback.TextColor = (Color)Resources["InputPlaceholderColor"];

Subject.Text = string.Empty;

Resources["PageBgColor"] = Resources["PageBgColorNoSubject"]; Resources["BtSubmitColor"] = Resources["BtInactiveColor"];

}



Hình 4-4 cho thấy kết quả cho cả hai nền tảng.

**Code đầy đủ: trang phản hồi sử dụng Resources động**

Danh sách 4-14 và danh sách 4-15 cung cấp Code hoàn chỉnh cho dynamicresourcepage. XAML và DynamicResourcePage.XAML.CS.

***Liệt kê 4-14.*** Phản hồi trang XAML với Resources động

***Hình 4-4.*** *Style động dựa trên tương tác người dùng*

<ContentPage BackgroundColor="{DynamicResource PageBgColor}"

Title="DynamicResource" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.DynamicResourcesPage">

<ContentPage.Resources>

<ResourceDictionary>

<x:String x:Key="SubjectPlaceholder">Enter Subject</x:String> <x:String x:Key="FeedbackPlaceholder">Type your message here </x:String>

<x:String x:Key="InputPlaceholder">Enter Subject</x:String> <Color x:Key="PageBgColor">#4CAF50</Color>

<Color x:Key="PageBgColorNoSubject">#85af87</Color>

<Color x:Key="PageBgColorWithSubject">#4CAF50</Color> <Color x:Key="FormBgColor">#AAAAAA</Color>

<Color x:Key="InputTextColor">Navy</Color>

<Color x:Key="InputPlaceholderColor">Silver</Color>

<Color x:Key="FeedbackBorderColor">Silver</Color> <FontAttributes x:Key="InputFontStyle">Italic</FontAttributes> <Color x:Key="BtTextColor">White</Color>

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color>

<Color x:Key="BtSubmitActiveColor">#4CAF50</Color>

<Color x:Key="BtCancelActiveColor">#f47442</Color>

<Color x:Key="BtInactiveColor">Gray</Color> <FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> </ResourceDictionary>

</ContentPage.Resources>

<Grid Padding="30" Margin="30" BackgroundColor="{DynamicResource FormBgColor}">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*" />

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions> <Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" /> </Grid.ColumnDefinitions> <Entry x:Name="Subject" Placeholder="{DynamicResource SubjectPlaceholder}"

Grid.Row="0" TextColor="{DynamicResource InputTextColor}" FontAttributes="{DynamicResource InputFontStyle}"

Unfocused="HandleSubject"

Focused="HandleSubject" />

<StackLayout Orientation="Vertical" Grid.Row="1" Padding="1"

BackgroundColor="{DynamicResource FeedbackBorderColor}"> <Editor x:Name="Feedback" VerticalOptions="FillAndExpand" TextColor="{DynamicResource InputTextColor}" FontAttributes="{DynamicResource InputFontStyle}" Unfocused="HandleFeedback" Focused="HandleFeedback" /> </StackLayout>

<StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Submit" HorizontalOptions="FillAndExpand" TextColor="{DynamicResource BtTextColor}" FontAttributes="{DynamicResource

BtFontStyle}" BackgroundColor="{DynamicResource BtSubmitColor}" />

<Button Text="Cancel" HorizontalOptions="FillAndExpand" TextColor="{DynamicResource BtTextColor}" FontAttributes="{DynamicResource

BtFontStyle}" BackgroundColor="{DynamicResource BtCancelColor}" />

</StackLayout>

</Grid>

</ContentPage>

***Liệt kê 4-15.*** Phản hồi Code trang đằng sau với Dynamic Resources

public partial class DynamicResourcesPage : ContentPage {

void HandleSubject(object sender, Xamarin.Forms.FocusEventArgs e) { if (Subject.Text == string.Empty){

Resources["PageBgColor"] = Resources["PageBgColorNoSubject"];

Resources["BtSubmitColor"] = Resources["BtInactiveColor"];

} else {

Resources["PageBgColor"] = Resources["PageBgColorWithSubject"];

Resources["BtSubmitColor"] = Resources["BtSubmitActiveColor"];

}

}

void HandleFeedback(object sender, Xamarin.Forms.FocusEventArgs e) { var text = Feedback.Text;

var placeHolderText = (string)Resources["FeedbackPlaceholder"];

if (Feedback.Text == placeHolderText) {

Feedback.Text = string.Empty;

Feedback.TextColor = (Color)Resources["InputTextColor"]; return;

}

if (Feedback.Text == string.Empty) {

Feedback.Text = placeHolderText;

Feedback.TextColor = (Color)Resources["InputPlaceholderColor"]; return;

}

}

public DynamicResourcesPage() {

InitializeComponent();

Feedback.Text = (string)Resources["FeedbackPlaceholder"]; Feedback.TextColor = (Color)Resources["InputPlaceholderColor"]; Subject.Text = "";

Resources["PageBgColor"] = Resources["PageBgColorNoSubject"]; Resources["BtSubmitColor"] = Resources["BtInactiveColor"];

}

}

DynamiCresource giúp chúng ta tạo nhiều giao diện người dùng tương tác hơn.

Quay lại Resources Dictionaries, xác định nó như là một tập tin XAML bên ngoài cho phép sử dụng lại trên toàn bộ ứng dụng, một kỹ thuật Code hóa tinh vi.

**Resources Dictionaries tái sử dụng**

Resources dictionaRies có thể được định nghĩa là riêng biệt, sử dụng lại XAML tài liệu có thể được tải vào một trang XAML hoặc bất cứ nơi nào trong đơn.

Trong ví dụ này, chúng tôi sẽ xác định Dictionaries Styleexample. MyResources Resource như là một tập tin XAML độc lập và tham chiếu trong importresourcespage. XAML bằng cách sử dụng các yếu tố resourcedictionary .

Tạo tài liệu XAML mới được gọi là Myresources. XAML và thay thế các phần tử gốc contentpage Visual Studio tạo với resourcedictionary cả XAML và Code sau.

Mẹo Xamarin. Forms 3.0 + sẽ không cần một Code phía sau, nhưng ví dụ này là Pre-3,0.

Sao chép Resources được xác định trong Dynamicresourcespage. XAML để myresources.

XAML, như được hiển thị trong danh sách 4-16.

***Liệt kê 4-16.*** Resources Dictionaries bên ngoài

<ResourceDictionary xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.MyResources">

<x:String x:Key="SubjectPlaceholder">Enter Subject</x:String> <x:String x:Key="FeedbackPlaceholder">Type your message here </x:String>

<x:String x:Key="InputPlaceholder">Enter Subject</x:String>

<Color x:Key="PageBgColor">#4CAF50</Color>

<Color x:Key="PageBgColorNoSubject">#85af87</Color>

<Color x:Key="PageBgColorWithSubject">#4CAF50</Color>

<Color x:Key="FormBgColor">#AAAAAA</Color>

<Color x:Key="InputTextColor">Navy</Color>

<Color x:Key="InputPlaceholderColor">Silver</Color>

<Color x:Key="FeedbackBorderColor">Silver</Color>

<FontAttributes x:Key="InputFontStyle">Italic</FontAttributes>

<Color x:Key="BtTextColor">White</Color>

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color>

<Color x:Key="BtSubmitActiveColor">#4CAF50</Color>

<Color x:Key="BtCancelActiveColor">#f47442</Color>

<Color x:Key="BtInactiveColor">Gray</Color>

<FontAttributes x:Key="BtFontStyle">Bold</FontAttributes>

</ResourceDictionary>

Bây giờ hãy tạo một tập tin mà sẽ tham chiếu Myresources. XAML. Sao chép dynamicresourcespage. XAML một trang mới được gọi là importresourcespage. XAML. Sử dụng xmlns để thêm không gian tên của myresources, StyleExamples, vào phần tử gốc contentpage .

xmlns:local="clr-namespace:StyleExamples"

Thay thế các resourcedictionary hiện tại importresourcespage. XAML với một yếu tố resourcedictionary duy nhất nhập myresources bằng cách sử dụng mergedwith tài sản, như thế này:

<ContentPage.Resources>

<ResourceDictionary MergedWith="local:MyResources"/>

</ContentPage.Resources>

Resources Dictionaries Myresources bây giờ có thể được duy trì tập trung và sử dụng trên toàn bộ ứng dụng.

**Global Resources Dictionaries**

Sau khi xác định một Resources Dictionaries có thể sử dụng lại, làm cho nó có sẵn rộng rãi bằng cách cấu hình App. XAML. Sửa đổi App. XAML, như được hiển thị trong danh sách 4-17.

***Liệt kê 4-17.*** Global Resources

<Application xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" xmlns:local="clr-

namespace:StyleExamples" x:Class="StyleExamples.App">

<Application.Resources>

<ResourceDictionary MergedWith="local:MyResources"/> </Application.Resources>

</Application>

Bây giờ bạn đã xác định các Resources Static, Dynamic and /or có khả năng sử dụng lại, Xamarin. Forms sẽ xác định những Resources nào để đăng ký tại thời gian chạy?

**Hành vi tra cứu**

Tìm kiếm Resources thời gian bắt đầu với dạng View, sau đó trang và sau đó tiến hành đối với các Resources ứng dụng khi cần thiết, vì vậy vị trí thực của Resources Dictionaries trong các vấn đề về ứng dụng. Đầu tiên, thời gian chạy tìm kiếmResources NY được xác định trong chế *View* đang thực hiện tìm kiếm. Sau đó, thời gian chạy đi qua cha mẹ tất cả các cách để *trang*. Nếu nó không thể tìm thấy bất cứ điều gì, sau đó các Resources xác định trong *ứng dụng* được tìm kiếm và cuối cùng là Resources *nền tảng* có chứa Resources mặc định để điều khiển.

Cố gắng giữ kích thước của Resources Dictionaries nhỏ. Sử dụng lại trong ngữ cảnh này đề cập đến việc viết một lần ở khắp mọi nơi nhưng không phải là một nguồn lực tồn tại chỉ một lần trong thời gian chạy. Thời gian tạo bản sao của Resources mỗi lần yêu cầu. Trường hợp nguồn lực được xác định cao hơn trong hệ thống phân cấp nhưng chỉ được sử dụng thêm xuống trong một trang là một dấu hiệu cho thấy Resources không cần thiết.

**Overriding Resources**

Khi bạn cần buộc sử dụng Resources khi có dạng View khác, PAGE hoặc các phiên bản ứng dụng của Resources này, bạn có thể ghi đè Resources. Xác định trong importresourcespage. XAML Resources Cục bộ tái sử dụng phím pagebgcolor để gán một Color khác và ghi đè lên một xác định trong Myresources. XAML, như được hiển thị trong danh sách 4-18 .

***Liệt kê 4-18.*** Nhập Resources Dictionaries bên ngoài sử dụng MergedWidth

<ContentPage.Resources>

<ResourceDictionary MergedWith="local:MyResources">

<Color x:Key="PageBgColor">Red</Color>

</ResourceDictionary>

</ContentPage.Resources>

Color đỏ được xác định tại địa phương là gán cho pagebgcolor sau khi myresources được nhập.

**Sáp nhập Dictionaries**

Resources có thể được tổ chức trong tài liệu XAML riêng biệt. Cách tiếp cận này làm giảm nỗ lực tìm kiếm và cải thiện hiệu suất nếu chỉ Dictionaries có liên quan đến một trang được tải.

Sử dụng phần tử tài sản Resourcedictionary. MergedDictionaries thay vì thuộc tính mergedwith để tải một hoặc nhiều Dictionaries bên ngoài, như được hiển thị trong danh sách 4-19.

***Liệt kê 4-19.*** Nhập nhiều Resources Dictionaries sử dụng MergedDictionaries

<ContentPage.Resources>

<ResourceDictionary.MergedDictionaries>

<local:MyResources/>

<!—other dictionaries here -->

</ResourceDictionary>

</ContentPage.Resources>

Dictionaries sáp nhập được tìm kiếm theo thứ tự chúng được liệt kê trong XAML. Resources có cùng khoá trong các Dictionaries tiếp theo ghi đè các Resources đã tuyên bố trước đó.

Resources là tuyệt vời để tránh các thuộc tính hardcoding trong ứng dụng. Tuy nhiên, sử dụng phần mở rộng đánh dấu làm cho Code XAML, làm cho nó ít có thể đọc được. Ngoài ra, mỗi Resources cần phải được gán riêng cho một sản. Đảm bảo một thiết kế phù hợp trở thành một thách thức. Tất cả điều này cũng có tác động hiệu suất. Tại thời gian chạy, mỗi Resources được nhìn cá nhân và được gán cho một thuộc tính của dạng View.

Nó sẽ không được tốt đẹp nếu thiết kế của một View cOuld được xác định một lần và cho phần còn lại của ứng dụng của bạn Xamarin. Forms sẽ chăm sóc của ảo thuật giao diện người dùng? Chào mừng bạn đến với thế giới của Styles.

**Style**

Style đòn bẩy Resources Dictionaries để cho phép chúng tôi tùy chỉnh giao diện của giao diện của chúng tôi trong một trang nhã, tái sử dụng Manner. Một *Style* là sự kết hợp của một hoặc nhiều setters thuộc tính được áp dụng cho một cái nhìn cụ thể nhắm mục tiêu theo Style. Kiểu được định nghĩa là Resources bên trong Resources Dictionaries. Như các Resources, Style có chức năng tương tự như các Resources khác, chẳng hạn như tra cứu Static và động, tái sử dụng thông qua Resources Dictionaries bên ngoài, ghi đè các kiểu địa phương xác định u p cao hơnvà sáp nhập các kiểu từ các nguồn khác nhau. Tất cả các dạng View thừa hưởng từ visualelement có thuộc tính Style có thể được sử dụng để tra cứu một kiểu bằng staticresource hoặc dynamicresource.

Tuy nhiên, Style có thể làm nhiều hơn: Style khi cung cấp một khóa là defined để nhắm mục tiêu một View cụ thể và được gọi là Explicit Style. Nếu không có một phím, một Style được gọi là một Implicit Styles và được áp dụng ngầm cho tất cả các phiên bản của dạng View được nhắm mục tiêu. Dạng View có thể ghi đè Values thuộc tính do kiểu cục bộ cung cấp, bằng cách gán Values mới cho thuộc tính. Thuộc tính setters xác định trong một Style có thể tra cứu các Resources khác trong Dictionaries để tránh trùng lặp XAML. Styles hỗ trợ thừa kế, cho phép Style để lấy được từ các Style khác. Cuối cùng, Xamarin. Forms cung cấp Device Styles tích hợp cho dạng View nhãn có thể được sử dụng trong ứng dụng để phản hồi động cài đặt trợ năng của người dùng.

Thận trọng các thuộc tính được sử dụng trong Style phải được tính chất tài sản Bindable và tồn tại trong View chính nó và không chỉ là lớp cơ sở của nó .

Xác định một Style nhắm mục tiêu Contentpage và chỉ định một setter bên trong Style gán Color cho thuộc tính Color nền của trang, như được hiển thị trong danh sách 4-20.

***Liệt kê 4-20.*** Thiết lập Color nền cho một trang nội dung

<ContentPage.Resources>

<ResourceDictionary>

<Style x:Key="PageBgColor" TargetType="ContentPage">

<Setter Property="BackgroundColor" Value="#4CAF50"/> </Style>

</ResourceDictionary>

</ContentPage.Resources>

Các thuộc tính Targettype là bắt buộc đối với một Style.

**Tra cứu Style**

Là một nguồn lực, các kiểu được nhìn lên bằng cách sử dụng hoặc là các phần mở rộng đánh dấu Staticresource hoặc dynamicresource . Tất cả các dạng View thừa hưởng từ visualelement có thuộc tính Style có thể được sử dụng để tra cứu một STYLe.

Gán một Style cho một Contentpage bằng cách sử dụng một trong hai staticresource, nếu Style sẽ không thay đổi trong suốt đời của ứng dụng.

< ContentPage Style = "{StaticResource PageBgColor}"/>

Hãy nhớ rằng, như với Staticresource, một Style phải tồn tại và tiếp theo chAnges của Style sẽ không được phản ánh.

Sử dụng Dynamicresource nếu Style sẽ thay đổi khi ứng dụng chạy.

< ContentPage Style = "{DynamicResource PageBgColor}"/>

Với Dynamicresource Values mặc định của các thuộc tính tài sản Bindable được sử dụng nếu kiểu bị thiếu. Style th e có thể được xác định tại thời gian chạy, và C# Code có thể được sử dụng để thay đổi Style tự động tại thời gian chạy.

**Explicit Style**

*Explicit Style* là Resources Style trong một Resources Dictionaries mà có một khóa, được sử dụng bởi một cái nhìn để tham khảo Style. Một Explicit Style không được áp dụng cho tất cả các phiên bản của khung nhìn nhắm mục tiêu. View phải Explicit tham chiếu nó trong thuộc tính kiểu của nó .

Để chỉ định một Explicit Style, thêm một yếu tố Style bên trong Resources Dictionaries của một trang, cho biết loại lượt View nó nhắm mục tiêu bằng cách sử dụng thuộc tính targettype , và gán một khóa cho Style để nó có thể được tham chiếu như một nguồn lực.

Tạo một trang mới được gọi là ExplicitStylePage. xAML và xác định một Style cụ thể thiết lập Color nền cho ExplicitStylePage.

1. Tạo một kiểu cụ thể cho một Button sử dụng kiểu buttonstyle, ví dụ:

< Style TargetType = "Button" x:Key = "ButtonStyle"/>

1. Bên trong yếu tố Style, cung cấp một hoặc nhiều yếu tố setter để gán Values cho các thuộc tính riêng lẻ. Xác định hai setter gán văn bản và Color nền, ví dụ:

<Setter Property="TextColor" Value="White" /> <Setter Property="BackgroundColor" Value="#4CAF50" />

1. Gán Style cho dạng View bằng cách sử dụng thuộc tính Style của View .

<Button Text="Styled Button" Style="{StaticResource ButtonStyle}"/>

Hình 4-5 cho thấy kết quả cho cả hai nền tảng.



***Hình 4-5.*** *Sử dụng Explicit Style*

**Code đầy đủ: sử dụng Explicit Styles**

Danh sách 4-21 cung cấp Code hoàn chỉnh cho ExplicitStylePage. XAML.

***Liệt kê 4-21.*** Xác định và sử dụng Explicit Style

<?xml version="1.0" encoding="UTF-8"?> <ContentPage Title="Styles Page" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.ExplicitStylePage">

<ContentPage.Resources> <ResourceDictionary>

<Style TargetType="Button" x:Key="ButtonStyle"> <Setter Property="TextColor" Value="White" /> <Setter Property="BackgroundColor" Value="#4CAF50" /> </Style>

</ResourceDictionary>

</ContentPage.Resources>

<StackLayout Orientation="Vertical">

<Button Text="Styled Button" Style="{StaticResource ButtonStyle}" />

<Button Text="Regular Button" />

</StackLayout>

</ContentPage>

**Implicit Styles**

*Implicit Styles* là Resources kiểu trong một Resources Dictionaries mà không có phím chỉ địnhEd và do đó được áp dụng cho tất cả các View chỉ định loại mục tiêu.

Tạo một trang XAML mới được gọi là Implicitstylepage. XAML và sao chép trang ExplicitStylePage. XAML. Loại bỏ các buttonstyle phím từ Style, như thế này, chỉ để lại targettype:

< Style TargetType = "Button"/>

Điều này sẽ áp dụng Style cho tất cả các Button được xác định trong contentpage. Ngoài ra, loại bỏ tất cả các Button và tạo một Button đơn giản mà không có Style, như thế này:

< Button Text = "Button Implicit Styles"/>

Hình 4-6 cho thấy kết quả cho cả hai nền tảng.



***Hình 4-6.*** *Sử dụng Implicit Styles*

**Code đầy đủ: sử dụng Implicit Styles**

Danh sách 4-22 cung cấp Code hoàn chỉnh cho implicitstylepage. XAML.

***Liệt kê 4-22.*** Xác định và sử dụng Implicit Styles

<?xml version="1.0" encoding="UTF-8"?>

<ContentPage Title="Implicit Styles" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.ImplicitStylePage">

<ContentPage.Resources>

<ResourceDictionary>

<Style TargetType="Button">

<Setter Property="TextColor" Value="White" />

<Setter Property="BackgroundColor" Value="#4CAF50" /> </Style>

</ResourceDictionary>

</ContentPage.Resources>

<StackLayout Orientation="Vertical">

<Button Text="Implicit Style Button"/>

</StackLayout>

</ContentPage>

Lưu ý loại mục tiêu kiểu phải khớp chính xác với loại giao diện đích. Nếu không, nó không được áp dụng. Ví dụ, một Style Visualelement sẽ không được áp dụng cho một Button ngầm.

Để áp dụng kiểu được xác định cho một lớp cơ sở của một dạng View cụ thể, một Style phải được đặt một cách Explicit. Xác định một Style cho dạng View lớp cơ sở , ví dụ:

<Style TargetType="View" x:Key="ViewStyle"/>

View không thể ngầm áp dụng cho Button. Một khóa phải được cung cấp để chỉ định Style một cách Explicit.

<Button Style="{StaticResource ViewStyle}"/>

**Style overriding**

Khi một View được instantiated, đầu tiên Style được áp dụng và sau đó bất kỳ Values địa phương gán cho các thuộc tính của nó. Điều này cho phép thay thế các Style tại địa phương, đó là thú vị khi đối phó với Style ngoại lệ. Bạn có thể ghi đè cả Explicit Style và tiềm ẩn.

Tạo một trang XAML gọi là Overridingstylespage. XAML. Sao chép ExplicitStylePage. XAML vào đó. Thêm một Button sử dụng buttonstyle nhưng có một định nghĩa địa phương của backgroundcolor, ví dụ:

<Button Text="Overriding Styled Button" Style="{StaticResource ButtonStyle}"

**BackgroundColor="Black"** />

Hình 4-7 cho thấy kết quả cho cả hai nền tảng.



***Hình 4-7. Các*** *Style overriding*

**Code đầy đủ: overriding Styles**

Danh sách 4-23 cung cấp Code hoàn chỉnh cho overridingstylespage. XAML.

***Liệt kê 4-23.*** Xác định và sử dụng Implicit Styles

<?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="StyleExamples.OverridingStylesPage">

<ContentPage.Resources>

<ResourceDictionary>

<Style TargetType="Button" x:Key="ButtonStyle">

<Setter Property="TextColor" Value="White" />

<Setter Property="BackgroundColor" Value="#4CAF50" />

</Style>

</ResourceDictionary>

</ContentPage.Resources>

<StackLayout Orientation="Vertical">

<Button Text="Styled Button" Style="{StaticResource

ButtonStyle}" />

<Button Text="Regular Button" />

<Button Text="Overriding Styled Button" Style="{StaticResource

ButtonStyle}"

BackgroundColor="Black"/>

</StackLayout>

</ContentPage>

**Sử dụng Resources trong Styles**

Nó là phổ biến cho các Style để chia sẻ cài đặt tài sản cho các quan điểm khác nhau. Các Button trong trang phản hồi chia sẻ cùng Color văn bản và Font Styles. Xác định các thuộc tính cho mỗi Style riêng kết quả trong XAML trùng lặp. Thuộc tính Values của Setter cho phép tìm kiếm Resources bằng cách sử dụng staticresource hoặc dynamicresource.

Tạo một tập tin XAML gọi Buttonstyleswithresourcespage. XAML với Resources Dictionaries. Xác định một Color và fontattributes như các Resources riêng biệt trong Dictionaries. Xác định hai Button Styles submitbutton và cancelbutton rằng cả hai tham chiếu các Resources bttextcolor và btfontstyle bằng cách sử dụng Phần mở rộng đánh dấu staticresource như được hiển thị trong danh sách 4-24.

***Liệt kê 4-24.*** Sử dụng Resources

<ResourceDictionary>

<Color x:Key="BtTextColor">Black</Color> <FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> <Style x:Key="SubmitButton" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

</Style>

<Style x:Key="CancelButton" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

</Style>

</ResourceDictionary>

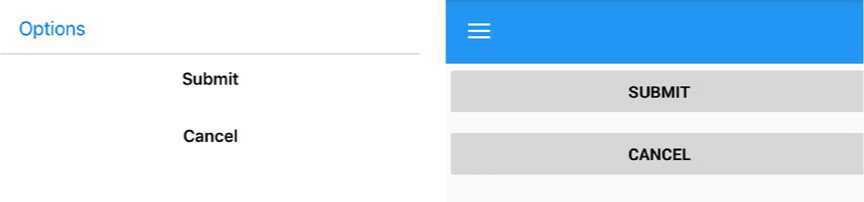
Phần mở rộng đánh dấu Dynamicresource cũng có thể được sử dụng, làm cho bttextcolor và btfontstyle tùy chọn tại thời gian thiết kế. Thời gian sử dụng Values mặc định cho các thuộc tính TextColor và fontattributes nếu các Resources không thể Found. Với staticresource Điều này dẫn đến lỗi thời gian chạy.

Bây giờ, áp dụng các kiểu này cho một Button Submit và Cancel trên trang, ví dụ:

<Button Text="Submit" Style="{StaticResource SubmitButton}" />

<Button Text="Cancel" Style="{StaticResource CancelButton}" />

Hình 4-8 Hiển thị các Button cho mỗi nền tảng.



***Hình 4-8.*** *Kiểu Button sử dụng Resources*

Cả hai kiểu, Submitbutton và hủy bỏ Button bây giờ chia sẻ Resources nhưng vẫn duy trì trùng lặp XAML Code để làm như vậy. Style thừa kế giải quyết vấn đề này.

**Hoàn thành Code: sử dụng Resources trong Styles**

Danh sách 4-25 Hiển thị buttonstyleswithresourcespage. XAML.

***Liệt kê 4-25.*** Buttonstylespage. XAML

<ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="http:// schemas.microsoft.com/winfx/2009/xaml" x:Class="StyleExamples.

ButtonStylesWithResourcesPage">

<ContentPage.Resources>

<ResourceDictionary>

<Color x:Key="BtTextColor">Black</Color>

<FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> <Style x:Key="SubmitButton" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

</Style>

<Style x:Key="CancelButton" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

</Style>

</ResourceDictionary> </ContentPage.Resources> <StackLayout Orientation="Vertical">

<Button Text="Submit" Style="{StaticResource SubmitButton}" />

<Button Text="Cancel" Style="{StaticResource CancelButton}" /> </StackLayout>

</ContentPage>

Trong ví dụ này, bạn có thể thấy rằng chúng tôi đang thiết lập cho cả hai Submitbutton và hủy bỏ các thuộc tính TextColor và fontattributes . Hãy kiểm tra làm thế nào chúng ta có thể tránh điều này thông qua Style thừa kế.

**Style thừa kế**

*Style thừa kế* cho phép một Style để SErve như là Style cơ bản của các Style bắt nguồn khác. Kiểu bắt nguồn có thể ghi đè hoặc thêm các thiết đặt thuộc tính.

Lưu ý một Style bắt nguồn chỉ có thể nhắm mục tiêu cùng hoặc một phân lớp của View nhắm mục tiêu theo Style cơ sở. Ngoài ra, một Implicit Styles có thể Lấy từ một Explicit Style , nhưng một Explicit Style không thể Lấy từ một Implicit Styles, bởi vì không có khoá được cung cấp cho một Implicit Styles.

Thừa kế được xử lý khác nhau tùy thuộc vào việc một Style có nguồn gốc Static hoặc tự động.

Hãy bắt đầu với định nghĩa Style kế thừa Static.

**Thừa kế Static Style**

*Kế thừa Style Static* được sử dụng để xác định kế thừa tại thời gian thiết kế. Sử dụng thuộc tính basedon để cung cấp tra cứu cho kiểu cơ sở.

Khi xác định một kiểu thừa kế Static, bạn tạo hai Style mà cả hai mục tiêu, ví dụ, một loại cụ thể như một Button. Tuy nhiên, Style thừa hưởng bắt nguồn từ Style đầu tiên bằng cách sử dụng thuộc tính basedon kết hợp với staticresource đánh dấu excăng thẳng, ví dụ:

<Style x:Key="Parent" TargetType="Button"/>

<Style x:Key="Child" TargetType="Button" BasedOn="{StaticResource Parent}"/>

Lưu ý chỉ có phần mở rộng đánh dấu Staticresource có thể được sử dụng với thuộc tính basedon .

Để chứng minh điều này, sao chép Buttonstyleswithresourcespage. XAML trang XAML mới gọi là buttonstyleswithstaticinheritancepage. XAML. Xác định một Style mới gọi là buttonstyle rằng cả hai submitbutton và hủy bỏ Button Lấy từ, như được hiển thị trong danh sách 4-26.

***Liệt kê 4-26.*** Kế thừa Static Style

<ResourceDictionary>

<Color x:Key="BtTextColor">Black</Color>

<FontAttributes x:Key="BtFontStyle">Bold</FontAttributes>

<Style x:Key="ButtonStyle" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

</Style>

<Style x:Key="SubmitButton" TargetType="Button" BasedOn="{StaticResource

ButtonStyle}" />

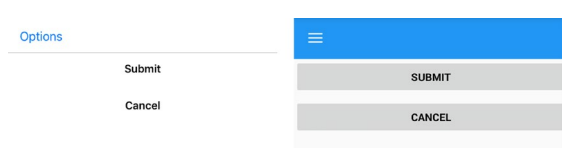
<Style x:Key="CancelButton" TargetType="Button"

BasedOn="{StaticResource

ButtonStyle}" />

</ResourceDictionary>

Các thay đổi chỉ có trong Resources Dictionaries. Các Button trông giống như hình 4-9 Hiển thị cho mỗi nền tảng.



***Hình 4-9.*** *Style Button với*  *kế thừa Style Static*

XAML trông sạch hơn bây giờ nhưng những gì nếu thừa kế có thể thay đổi tại thời gian chạy? Sử dụng kế thừa Style động trong trường hợp này.

**Dynamic Style thừa kế**

*Dynamic Style thừa kế* định kiểu kế thừa tại thời gian chạy. Sử dụng thuộc tính baseresourcekey để chỉ định tên của Style cơ sở, có thể được cung cấp tại thời gian chạy.

Khi xác định một Style động thừa kế, tạo ra hai Style mà cả hai mục tiêu một loại yếu tố cụ thể như buttton, cha mẹ, và một Style con. Style thừa hưởng con sử dụng thuộc tính baseresourcekey để tham khảo tên của kiểu cha mẹ mà không sử dụng phần mở rộng đánh dấu staticresource để tra cứu Resources, ví dụ:

<Style x:Key="Parent" TargetType="Button"/>

<Style x:Key="Child" TargetType="Button" BaseResourceKey="Parent"/>

Lưu ý Baseresourcekey không sử dụng tra cứu Dictionaries.

Style Parent không cần phải tồn tại trong thời gian thiết kế. Thời gian chạy sẽ sử dụng các thiết lập mặc định của một lượt View khi trongstantiating Style con, nếu không có cha mẹ có thể được tìm thấy.

Để chứng minh điều này, sao chép Buttonstyleswithstaticinheritancepage. XAML trang XAML mới gọi là buttonstyleswithdynamicinheritancepage. XAML. Thay thế các thuộc tính basedon của submitbutton và có thểcelbutton với thuộc tính baseresourcekey cho biết kế thừa Style động, như được hiển thị trong Liệt kê 4-27.

***Liệt kê 4-27.*** Dynamic Style thừa kế

<ResourceDictionary>

<Color x:Key="BtTextColor">Black</Color> <FontAttributes x:Key="BtFontStyle">Bold</FontAttributes>

<Style x:Key="ButtonStyle" TargetType="Button">

<Setter Property="TextColor" Value="{StaticResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{StaticResource BtFontStyle}" />

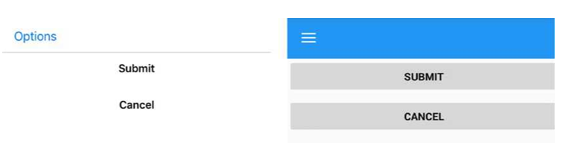
</Style>

<Style x:Key="SubmitButton" TargetType="Button" BaseResourceKey="ButtonStyle" />

<Style x:Key="CancelButton" TargetType="Button" BaseResourceKey="ButtonStyle" />

</ResourceDictionary>

Một lần nữa, những thay đổi chỉ có trong Resources Dictionaries. Các Button trông giống như hình 4-10 Hiển thị cho mỗi nền tảng.



***Hình 4-10. Kiểu*** *Button với Style động thừa kế*

Một Style mới có thể được chỉ định trong Code sau.

Resources ["SubmitButton"] = Resources ["InactiveButton"];

**Tổng quan về Style**

Hãy View lại trang phản hồi và áp dụng các Style khác nhau được thảo luận như là thừa kế kiểu, Implicit Styles và Explicit, sử dụng Resources và định nghĩa các Style như Dictionaries bên ngoài. Trước tiên, chúng tôi sẽ tạo một tập tin Resources Dictionaries xác định Style của chúng ta bằng cách sử dụng các kỹ thuật được thảo luận trong chương này và sau đó áp dụng các kiểu này cho một trang phản hồi và kiểm tra cách kết quả cuối cùng so sánh với trang phản hồi ban đầu .

Tạo một trang XAML Mystyles. XAML và sao chép myresources. XAML . Thêm các kiểu sau vào Dictionaries:

1. tạo một Implicit Styles cho View contentpage thiết lập Color nền cho Resources pagebgcolor .

2. Entry và Editor có một inputview lớp Bas e nhưng không chia sẻ nhiều thuộc tính ở cấp lớp cơ sở. Tạo một kiểu cơ sở gọi là inputviewstyle cho inputview đặt verticaloptions thành fillandexpand. Xác định hai kiểu tiềm ẩn cho mục nhập và trình soạn thảo kế thừa từ inputviewstyle và đặt cho mỗi TextColor với inputtextcolor và fontattributes để inputfontstyle. Mục cũng thêm một setter cho trình Placeholder để tra cứu văn bản từ Resources chính.

3. thêm một setter vào buttonstyle cho horizontaloptions và setters cho backgroundcolor và văn bản cho submitbutton và kiểu Button hủy .

4. tạo một Implicit Styles cho bố cục Grid với mặc định padding , marginvà backgroundcolor.

5. tạo một Explicit Style cho stacklayout được gọi là Feedbackborder được sử dụng xung quanh biên tập để cung cấp một biên giới.

Code đầy đủ: xác định kiểu

Danh sách 4-28 Hiển thị mystyles. XAMLCập Nhật. Các Resources được bao gồm trong myresources. XAML (danh sách 4-16).

***Liệt kê 4-28.*** Mystyles. XAML

<ResourceDictionary xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.MyStyles">

<Style TargetType="ContentPage">

<Setter Property="BackgroundColor" Value="{DynamicResource

PageBgColor}" />

</Style>

<Style x:Key="InputViewStyle" TargetType="InputView">

<Setter Property="VerticalOptions" Value="FillAndExpand" /> </Style>

<Style TargetType="Entry" BaseResourceKey="InputViewStyle">

<Setter Property="FontAttributes" Value="{DynamicResource InputFontStyle}" />

<Setter Property="TextColor" Value="{DynamicResource InputTextColor}" />

<Setter Property="Placeholder" Value="{DynamicResource

SubjectPlaceholder}" />

</Style>

<Style TargetType="Editor" BaseResourceKey="InputViewStyle">

<Setter Property="FontAttributes" Value="{DynamicResource InputFontStyle}" />

<Setter Property="TextColor" Value="{DynamicResource

InputTextColor}" />

</Style>

<Style x:Key="ButtonStyle" TargetType="Button">

<Setter Property="HorizontalOptions" Value="FillAndExpand" /> <Setter Property="TextColor" Value="{DynamicResource BtTextColor}" />

<Setter Property="FontAttributes" Value="{DynamicResource

BtFontStyle}" />

</Style>

<Style x:Key="SubmitButton" TargetType="Button"

BaseResourceKey="ButtonStyle">

<Setter Property="BackgroundColor" Value="{DynamicResource BtSubmitColor}" />

<Setter Property="Text" Value="Submit" />

</Style>

<Style x:Key="CancelButton" TargetType="Button" BaseResourceKey="ButtonStyle">

<Setter Property="BackgroundColor" Value="{DynamicResource BtCancelColor}" />

<Setter Property="Text" Value="Cancel" />

</Style>

<Style TargetType="Grid">

<Setter Property="Padding" Value="30" />

<Setter Property="Margin" Value="30" />

<Setter Property="BackgroundColor" Value="{DynamicResource FormBgColor}" />

</Style>

<Style x:Key="FeedbackBorder" TargetType="StackLayout">

<Setter Property="Orientation" Value="Vertical" />

<Setter Property="BackgroundColor" Value="{DynamicResource FeedbackBorderColor}" />

<Setter Property="Padding" Value="1" />

</Style>

<x:String x:Key="SubjectPlaceholder">Enter Subject</x:String>

<x:String x:Key="FeedbackPlaceholder">Type your message here </x:String>

<x:String x:Key="InputPlaceholder">Enter Subject</x:String>

<Color x:Key="PageBgColor">#4CAF50</Color>

<Color x:Key="PageBgColorNoSubject">#85af87</Color>

<Color x:Key="PageBgColorWithSubject">#4CAF50</Color>

<Color x:Key="FormBgColor">#AAAAAA</Color>

<Color x:Key="InputTextColor">Navy</Color>

<Color x:Key="InputPlaceholderColor">Silver</Color>

<Color x:Key="FeedbackBorderColor">Silver</Color> <FontAttributes x:Key="InputFontStyle">Italic</FontAttributes>

<Color x:Key="BtTextColor">White</Color>

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color>

<Color x:Key="BtSubmitActiveColor">#4CAF50</Color>

<Color x:Key="BtCancelActiveColor">#f47442</Color>

<Color x:Key="BtInactiveColor">Gray</Color>

<FontAttributes x:Key="BtFontStyle">Bold</FontAttributes> </ResourceDictionary>

Đối với các Style có hiệu lực, chúng tôi loại bỏ tất cả các thuộc tính thiết lập trực tiếp và thay vì sử dụng thuộc tính Style để áp dụng Style của chúng tôi từ mystyles. XAML. Áp dụng các kiểu cho trang của bạn:

1. Loại bỏ các thuộc tính Backgroundcolor gán cho contentpage. Điều này được thiết lập bây giờ ngầm.

2. Xoá trình Placeholder, textcolor và gán thuộc tính fontattributes từ mục View. Đây là thiết lập bây giờ ngầm.

3. Loại bỏ verticaloptions, textcolor và các thuộc tính fontattributes từ View Editor . Đây là thiết lập bây giờ ngầm.

4. Loại bỏ horizontaloptions, textcolor, fontattributes và cài đặt thuộc tính backgroundcolor từ hai Button. Thay vì sử dụng thuộc tính Style của các Button để tra cứu submitbutton và cancelbutton, tương ứng.

5. Xoá các thiết đặt thuộc tính padding, marginvà backgroundcolor từ Grid. Đây là bây giờ đặt ngầm.

6. Loại bỏ các thiết đặt thuộc tính padding và backgroundcolor từ stacklayout bao gồm các trình soạn thảo. Thay vào đó đặt thuộc tính kiểu stacklayout Tìm kiếm kiểu feedbackborder.

**Code đầy đủ: áp dụng Style**

Danh sách 4-29 Hiển thị kết quả stylespage. XAML. Các định nghĩa về cột và hàng Grid được bao gồm trong danh sách 4-9.

***Liệt kê 4-29.*** Stylespage. XAML

<ContentPage Title="Styles Page" xmlns:local="clr-namespace:StyleExamples" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.StylesPage">

<ContentPage.Resources>

<ResourceDictionary MergedWith="local:MyStyles"/> </ContentPage.Resources>

<ContentPage.Content>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*" />

<RowDefinition Height="Auto"/> </Grid.RowDefinitions> <Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" /> </Grid.ColumnDefinitions>

<Entry x:Name="Subject" Grid.Row="0" Focused="HandleSubject" Unfocused="HandleSubject" />

<StackLayout Style="{StaticResource FeedbackBorder}" Grid. Row="1">

<Editor x:Name="Feedback" Focused="HandleFeedback" Unfocused="HandleFeedback" />

</StackLayout>

<StackLayout Orientation="Horizontal" Grid.Row="2"> <Button Style="{StaticResource SubmitButton}" /> <Button Style="{StaticResource CancelButton}" />

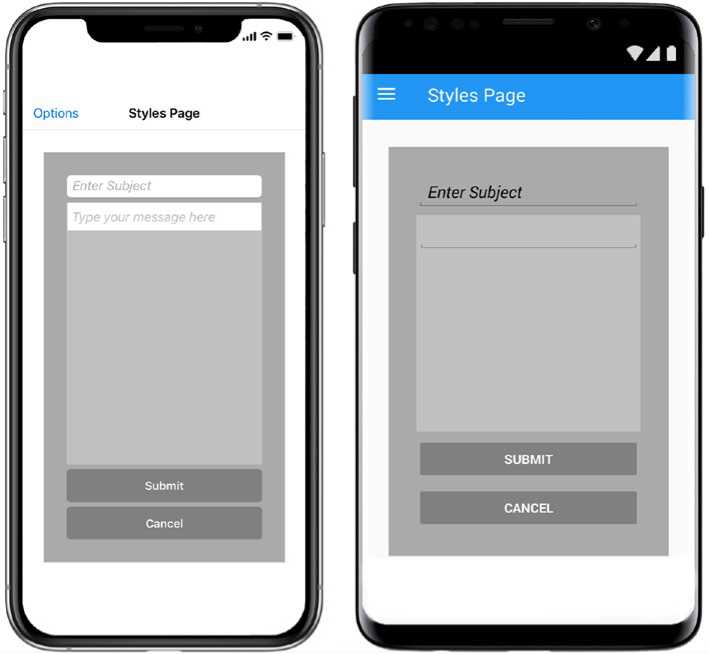
</StackLayout>

</Grid>

</ContentPage.Content>

</ContentPage>

Hình 4-11 cho thấy kết quả cho cả hai nền tảng.



***Hình 4-11.*** *Trang phản hồi sử dụng kiểu*

So sánh Stylespage. XAML với bản gốc Feedbackpage. XAML trong danh sách 4-1. Họ là gần như giống hệt nhau. Stylespage chỉ có một vài thiết đặt thuộc tính bổ sung cho các Button và stacklayout để bù đắp cho việc thiếu một thuộc tính biên giới cho trình soạn thảo .

Sử dụng kiểu loại bỏ sự lộn xộn từ XAML và làm cho nó có thể duy trì và nhất quán trên toàn bộ ứng dụng.

Xamarin. Forms cung cấp Device Styles cho nhãn để cho phép ứng dụng trả lời các thiết đặt trợ năng của người dùng. Hãy khám phá tiếp theo.

**Device Styles**

Xamarin. Forms cung cấp cho kiểm soát nhãn sáu kiểu được Constructor trong điều chỉnh tự động đến cài đặt trợ năng của thiết bị, gọi là *Device Styles*:

1. TitleStyle
2. SubTitleStyle
3. BodyStyle Style
4. CaptionStyle
5. ListItemTextStyle
6. ListItemDetailTextStyle

DeVice Styles có thể được sử dụng như bất kỳ Style nào khác trực tiếp trên nhãn View bằng cách sử dụng thuộc tính Style , ví dụ:

< Label Style = "{DynamicResource TitleStyle}" >

Device Styles cũng có thể phục vụ như là kiểu cơ sở cho các dạng View nhãn khác bằng cách sử dụng baseresourcekey của một yếu tố Style , ví dụ:

<Style x:Key="MyTitleStyle" TargetType="Label"

BaseResourceKey="TitleStyle">

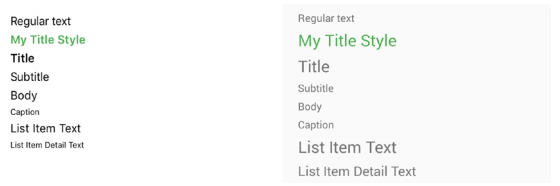
<Setter Property="TextColor" Value="#4CAF50" />

</Style>

Lựa chọn thứ hai là thú vị vì nó cho phép một ứng dụng để đáp ứng các thiết lập khả năng tiếp cận của một người sử dụng.

Lưu ý sử dụng Dynamicresource để tìm kiếm kiểu để đảm bảo View được Cập Nhật nếu người dùng thay đổi các thiết đặt trợ năng.

Hình 4-12 cho thấy cách các Device Styles tích hợp được hiển thị trên iOS và Android.



***Hình 4-12.*** *Sử dụng Device Styles*

**Hoàn thành Code: sử dụng Device Styles**

Danh sách 4-30 cung cấp Code hoàn chỉnh cho DeviceStylePage. XAML.

***Liệt kê 4-30.*** Stylespage. XAML

<ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.DeviceStylesPage">

<ContentPage.Resources>

<ResourceDictionary>

<Style x:Key="MyTitleStyle" TargetType="Label" BaseResourceKey="TitleStyle">

<Setter Property="TextColor" Value="#4CAF50" /> </Style>

</ResourceDictionary>

</ContentPage.Resources>

<StackLayout Padding="20,20">

<Label Text="Regular text" />

<Label Text="My Title Style" Style="{DynamicResource

MyTitleStyle}" />

<Label Text="Title" Style="{DynamicResource TitleStyle}" />

<Label Text="Subtitle" Style="{DynamicResource SubtitleTextStyle}" />

<Label Text="Body" Style="{DynamicResource BodyStyle}" />

<Label Text="Caption" Style="{DynamicResource CaptionStyle}" /> <Label Text="List Item Text" Style="{DynamicResource ListItemTextStyle}" />

<Label Text="List Item Detail Text" Style="{DynamicResource ListItemDetailTextStyle}" />

</StackLayout>

</ContentPage>

Trang phản hồi được xác định kiểu cho các điều khiển được chọn làm Resources Dictionaries có sử dụng lại. Tạo một bộ sưu tập các Style được thiết kế và gắn kết tốt là một nghệ thuật trong và của chính nó. Một đường cơ sở để bắt đầu từ là Themes được availaBLE cho Xamarin. Forms.

**Themes**

Các *Themes* xamarin. Forms là tập hợp các Style cho các yếu tố hình ảnh phổ biến và bao gồm các tùy chọn kiểu cho các điều khiển tiêu chuẩn. Themes là một cách tuyệt vời để tìm hiểu về Style, và họ cung cấp hai mẫu làm sẵn. Nhiều nền tảng hỗ trợ một chế độ Light và tối. Chế độ Light provides tương phản cao và rất hữu ích trong thời gian ngày. Đối với điều kiện Light yếu hoặc làm việc vào đêm, một chế độ tối có thể dễ chịu hơn. Xamarin. tạo thành một gói nuget Themes sáng và tối có thể được thêm vào một dự án để thực hiện một Light hoặc một Dark theme.

Sử dụngDark theme hoặc Light của t, giúp khởi động việc tạo ra các Tùy chỉnh Themes của riêng bạn, trong khi bạn chỉ ghi đè các kiểu nếu cần.

Lưu ý các Themes đã được View trước trong nhiều năm nay và được sử dụng tốt nhất như một công cụ học tập cho các Style, trừ khi chúng được phát hành để stable.

**Sử dụng Themes**

Sử dụng Themes là một quá trình bốn bước:

1. Chuyển nhập các gói NuGet Themes Light hoặc Dark.
2. Khởi tạo Themes trong các dự án dành riêng cho nền tảng.
3. Thêm không gian tên Themes trong trang hoặc ứng dụng.
4. Tải Themes làm Resources Dictionaries.

Code hóa để sử dụng Themes sáng và tối là gần như giống hệt nhau. Hãy bắt đầu với Theme Dark cho ví dụ chính của chúng ta và sau đó giải quyết Theme Light ở cuối. Dưới đây là bốn bước chi tiết:

1. Nhập Themes Color tối vào dự án của bạn bằng cách thêm xamarin. Forms. Theme. Base nuget gói cũng như các gói tối tương ứng cho cả hai nền tảng cụ thể dự án (ios và Android).

1. Themes vẫn còn trong bản View trước và đôi khi không tải đúng do lỗi. Sử dụng các bước sau trong từng dự án dành riêng cho nền tảng để khởi tạo chúng:

Trong dự án dành riêng cho nền tảng ios, thêm các dòng Code sau khi phương pháp loadapplication trong tệp appdelegate.CS :

Var x = typeof (Xamarin. Forms. Themes. Darkthemeresources);

X = typeof (Xamarin. Choms. Themes. Ios. Underlineeffect);

Tương tự, trong dự án dành riêng cho nền tảng Android, thêm các dòng Code sau phương pháp loadapplication trong tệp mainactivity.CS :

Var x = typeof (xamarin. Forms. Themes. Darkthemeresources); x = typeof (Xamarin. Forms. Themes. Android. Underlineeffect);

1. Thêm không gian tên App. XAML bằng tên Xamarin. Forms. Themes và chi tiết xamarin. Forms. Theme. Tối tăm, ví dụ:

Xmlns: Theme = "CLR-tênkhông gian: Xamarin. Forms. Themes; hội = xamarin. Forms. Theme. Dark"

1. Tải một Dark theme như một Resources Dictionaries bên ngoài trong App. XAML để sử dụng các Themes trên Global, như được hiển thị trong danh sách 4-31.

***Liệt kê 4-31.*** Sử dụng Theme Dark

<Application xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="mynamespace.App" xmlns:theme="clr- namespace:Xamarin.Forms.Themes;assembly=Xamarin.Forms.Theme.Dark">

<Application.Resources>

<ResourceDictionary MergedWith="theme:DarkThemeResources" />

</Application.Resources>

</Application>

Mẹo sử dụng Light theme sau quá trình tương tự. Chỉ cần thay thế từ "Dark" với "Light" và bạn là tốt để đi. thêm về điều này sớm.

Hãy thêm một số tùy chọn tạo kiểu cho các điều khiển được hỗ trợ để hiển thị những gì các Themes được tất cả về.

**Tùy chọn Themes Style**

Cả hai Themes light và dark cung cấp tùy chọn kiểu cho các điều khiển được chọn. Tùy chọn S tyle có sẵn cho các điều khiển sau:

• **Boxview Style:**  Horizontalrule, vòng tròn, và tròn.

• **Kiểu hình ảnh:**  tròn, tròn, và thumbnail. Làm tròn và hình thu nhỏ hiện đang hoạt động như nhau.

• **Style Button:**  mặc định, tiểu học, thành công, thông tin, cảnh báo, nguy hiểm, liên kết, nhỏ, và lớn.

• **Kiểu nhãn:**  tiêu đề, tiêu đề phụ, thân, liên kết, và nghịch đảo.

Để áp dụng tùy chọn kiểu cho dạng View, hãy sử dụng thuộc tính Styleclass của dạng View . Ví dụ: nếu bạn muốn thêm tiêu đề vào ứng dụng của mình, hãy đặt styleclass vào tiêu đề tùy chọn kiểu trên nhãn của bạn.

<Label Text="Header" StyleClass="Header"/>

Hãy View các tùy chọn Style như thế nào trong dark và Themes light.

**Dark Theme**

Hãy tạo một trang sử dụng tất cả các tùy chọn kiểu trong Dark theme có sẵn cho Button và nhãn. Tạo một trang có tên là darkthemepage. XAML và nhập Dark theme, như được hiển thị trong danh sách 4-32. Thêm vào trang chín Button và năm nhãn, mỗi hãng sử dụng một trong các tùy chọn tạo kiểu trước.

***Liệt kê 4-32.*** Trang Dark theme

<ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.DarkThemePage" xmlns:theme="clr- namespace:Xamarin.Forms.Themes;assembly=Xamarin.Forms.Theme.Dark"> <ContentPage.Resources>

<ResourceDictionary MergedWith="theme:DarkThemeResources" />

</ContentPage.Resources>

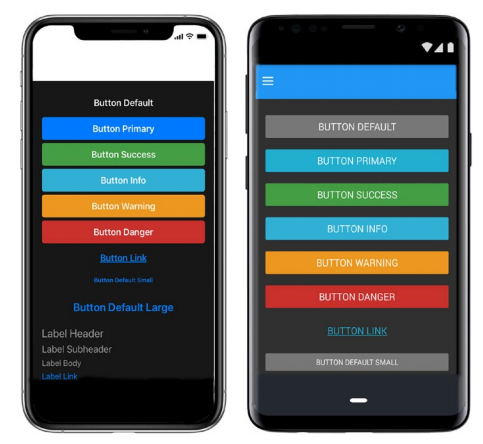
<StackLayout Padding="20">

<Button Text="Button Default" StyleClass="Default" /> <Button Text="Button Primary" StyleClass="Primary" /> <Button Text="Button Success" StyleClass="Success" /> <Button Text="Button Info" StyleClass="Info" /> <Button Text="Button Warning" StyleClass="Warning" /> <Button Text="Button Danger" StyleClass="Danger" /> <Button Text="Button Link" StyleClass="Link" /> <Button Text="Button Default Small" StyleClass="Small" /> <Button Text="Button Default Large" StyleClass="Large" /> <Label Text="Label Header" StyleClass="Header" /> <Label Text="Label Subheader" StyleClass="Subheader" /> <Label Text="Label Body" StyleClass="Body" /> <Label Text="Label Link" StyleClass="Link" /> <Label Text="Label Inverse" BackgroundColor="White" StyleClass="Inverse"/>

</StackLayout>

</ContentPage>

Hình 4-13 Hiển thị các tùy chọn kiểu có sẵn cho Dark theme trên iOS và Android.



***Hình 4-13.*** *Dark theme với các tùy chọn kiểu có sẵn*

**Light Theme**

Bây giờ cho Light theme , trước tiên bạn cần thực hiện bốn bước chúng ta đã sử dụng trước đó cho "Dark Theme" dưới "sử dụng Themes" để khởi tạo Theme Light, thay thế "Dark" với "Light".

Tiếp theo tạo một trang gọi là Lightthemepage. XAML và nhập Light theme , như được hiển thị trong danh sách 4-33. Thêm vào trang cùng chín Button và năm nhãn, mỗi người sử dụng một trong các tùy chọn.

***Liệt kê 4-33.*** Trang Light theme

<ContentPage xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.LightThemePage" xmlns:theme="clr- namespace:Xamarin.Forms.Themes;assembly=Xamarin.Forms.Theme.Light"> <ContentPage.Resources>

<ResourceDictionary MergedWith="theme:LightThemeResources" />

</ContentPage.Resources>

<StackLayout Padding="20">

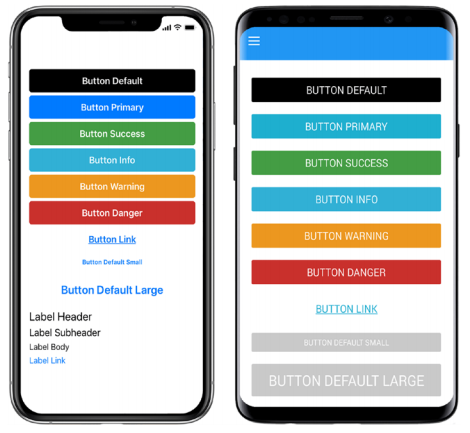
<Button Text="Button Default" StyleClass="Default" BackgroundColor="Black"/>

<Button Text="Button Primary" StyleClass="Primary" /> <Button Text="Button Success" StyleClass="Success" /> <Button Text="Button Info" StyleClass="Info" /> <Button Text="Button Warning" StyleClass="Warning" /> <Button Text="Button Danger" StyleClass="Danger" /> <Button Text="Button Link" StyleClass="Link" /> <Button Text="Button Default Small" StyleClass="Small" /> <Button Text="Button Default Large" StyleClass="Large" /> <Label Text="Label Header" StyleClass="Header" /> <Label Text="Label Subheader" StyleClass="Subheader" /> <Label Text="Label Body" StyleClass="Body" /> <Label Text="Label Link" StyleClass="Link" /> <Label Text="Label Inverse" BackgroundColor="White" StyleClass="Inverse"/>

</StackLayout>

</ContentPage>

Light theme với các tùy chọn kiểu được áp dụng giống như hình 4-14.



***Hình 4-14.*** *Themes sáng với các tùy chọn kiểu có sẵn*

**LIÊN KẾT BUTTON**

BUTTON MẶC ĐỊNH NHỎ

**Tùy chỉnh Themes**

Tùy chỉnh Themes là Themes bạn xác định dựa trên Dark theme và Light. Sau khi sử dụng một trong hai Themes, tùy chỉnh nó. Xác định một Style có nguồn gốc từ tùy chọn Style bằng cách sử dụng thuộc tính lớp của yếu tố kiểu và ghi đè các thuộc tính của tùy chọn kiểu với một hoặc nhiều cài đặt thuộc tính, ví dụ:

<Style TargetType="Label" Class="Header">

<Setter Property="TextColor" Value="Red"/>

</Style>

Bây giờ, chúng ta hãy trở lại trang phản hồi ban đầu và xác định một Tùy chỉnh Themes để Style nó.

**Tùy chỉnh Dark Theme**

Tạo một tệp Resources Dictionaries mới được gọi là Customtheme. XAML và ghi đè lên sự thành công và kiểu cảnh báo cho Button btsubmitcolor và BtCancelColor, như được hiển thị trong danh sách 4-34.

***Liệt kê 4-34.*** Địnhnghĩa Tùy chỉnh Themes

<ResourceDictionary xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.CustomTheme">

<Color x:Key="BtSubmitColor">#4CAF50</Color>

<Color x:Key="BtCancelColor">#f47442</Color>

<Style TargetType="Button" Class="Success">

<Setter Property="BackgroundColor" Value="{StaticResource B **tSubmitColor** }"/>

</Style>

<Style TargetType="Button" Class="Warning">

<Setter Property="BackgroundColor" Value="{StaticResource B **tCancelColor** }"/>

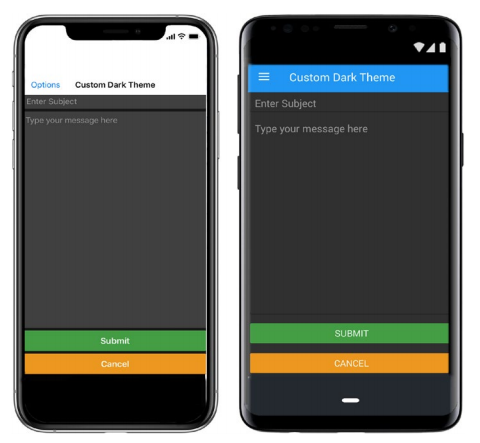
</Style>

</ResourceDictionary>

Nhân đôi Feedbackpage. XAML và gọi mới File customdarkthemepage. XAML. Tải Tùy chỉnh Themes vào customdarkthemepage. Gán các Style thành công và cảnh báo cho các Button gửi và hủy, ví dụ:

<Button Text="Cancel" StyleClass="Warning" /> <Button Text="Submit" StyleClass="Success" />

Hình 4-15 Hiển thị Dark theme tùy chỉnh được áp dụng cho trang phản hồi.



***Hình 4-15.*** *Dark theme tùy chỉnh được áp dụng cho trang phản hồi*

**Hoàn thành Code: tùy chỉnh Themes Color tối**

Danh sách 4-35 cung cấp Code hoàn chỉnh cho customdarkthemepage. XAML.

***Liệt kê 4-35.*** Customdarkthemepage. XAML

<ContentPage Title="Custom Dark Theme" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" x:Class="StyleExamples.CustomDarkThemePage" xmlns:theme="clr- namespace:Xamarin.Forms.Themes;assembly=Xamarin.Forms.Theme.Dark">

<ContentPage.Resources>

<ResourceDictionary MergedWith="theme:DarkThemeResources" />

</ContentPage.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Entry Placeholder="Enter Subject" Grid.Row="0" />

<Editor x:Name="Feedback" Grid.Row="1" Unfocused="HandleFeedback" Focused="HandleFeedback" />

<StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Cancel" StyleClass="Warning" />

<Button Text="Submit" StyleClass="Success" />

</StackLayout>

</Grid>

</ContentPage>

Cho đến bây giờ, cú pháp XAML được sử dụng để định kiểu ứng dụng. Một cú pháp thay thế và phổ biến cho các trang web là Cascading Style Sheets, được hỗ trợ bởi Xamarin. Các hình thức là tốt.

**Cascading Style Sheets (CSS)**

*Cascading Style Sheets* (CSS) là một ngôn ngữ để mô tả trình bày của các trang web ngôn ngữ đánh dấu Hypertext (HTML). Xamarin. Forms đã thông qua ký hiệu này như là một thay thế cho cú pháp XAML. Trong nội bộ, CSS được phân tích cú pháp và các động cơ Style tương tự được sử dụng để áp dụng thiết kế.

CSS bao gồm một bộ quy tắc, mỗi bắt đầu với một hoặc nhiều *Selectors* phân cách bằng dấu phẩy và theo sau bởi một hoặc nhiều cặp Values thuộc tính bên trong niềng răng xoăn.

Mẹo bất cứ điều gì bạn có thể làm với Style XAML, bạn có thể làm với CSS và ngược lại.

Thận trọng sử dụng một trong hai kiểu hoặc CSS nhưng không phải là sự kết hợp của cả hai khi phát triển một ứng dụng để đơn giản hóa việc khắc phục sự cố.

**Selectors**

Một bộ *chọn* điểm đến phần tử XAML mà cài đặt thuộc tính áp dụng. Giải phẫu cơ bản của một quy tắc CSS được hiển thị trong danh sách 4-36.

***Liệt kê 4-36.*** Quy tắc CSS

Selector {

Property1: Value1;

Property2: Value2;

}

Nếu tìm thấy một yếu tố kết hợp XAML, sau đó các Values được gán cho các thuộc tính tương ứng, chẳng hạn như Value1 để Property1 và Value2 để Property2. Điều này rất giống với một yếu tố kiểu XAML và setters. Một selector xác định loại trang, bố cục, hoặc điều khiển và có thể là mối quan hệ giữa chúng mà cần phải được hài lòng trước khi một Style được áp dụng.

Danh sách các Values thuộc tính và bộ chọn được hỗ trợ được cung cấp ở cuối phần dưới dạng tham chiếu. Selectors có thể được kết hợp mà không có bất kỳ giới hạn. Tất cả các kiểu phù hợp được áp dụng theo thứ tự của định nghĩa. Styles áp dụng trên chính điều khiển được áp dụng cuối cùng. Điều này được gọi là tầng, có nghĩa là trong ngữ cảnh này là quy tắc cụ thể nhất được áp dụng cuối cùng .

Lưu ý tên bộ chọn là trường hợp nhạy cảm. tên tài sản phải là trường hợp thấp hơn. Values tài sản không trường hợp nhạy cảm.

Hãy View lại trang phản hồi gốc và sử dụng CSS để tạo kiểu cho nó.

**Sử dụng CSS**

Sử dụng ký hiệu CSS cho các yếu tố Xamarin. Forms kiểu. Kiểu lớp cụ thể với . StyleClassName ký hiệu, quan điểm với #ViewName, và các loại lớp với một className.

Hãy Style ứng dụng phản hồi của chúng tôi bằng cách sử dụng CSS. Trước tiên, bạn sẽ cần một tập tin CSS. Tạo một Style trang được gọi là Styles. CSS trong dự án và sử dụng các hành động Constructor để xác định các tập tin như một embeddedresource.

Chúng ta muốn xác định Style Button Submit của chúng ta với một font chữ d Color trắng và Bolvà một Color nền Color xanh lá cây. Đối với điều này, hãy sử dụng ký hiệu chọn. Styleclassname để xác định một quy tắc CSS cho lớp kiểu submitbutton . Sử dụng Color thuộc tính cho TextColor, Font Styles cho fontattributes và Color nền cho backgroundcolor để chỉ định các Values, như được hiển thị trong danh sách 4-37.

***Liệt kê 4-37.*** Quy tắc CSS cho lớp kiểu SubmitButton

.SubmitButton {

color: white;

font-style: bold; background-color: #4CAF50;

}

Lặp lại tương tự cho Buttonloại bỏ kiểu, xác định bộ chọn nhưng thay vào đó sử dụng Color nền cam của chúng tôi, như được hiển thị trong danh sách 4-38.

***Liệt kê 4-38.*** CSS Rule cho Class CancelButton

.CancelButton {

color: white;

font-style: bold; background-color: #f47442;

}

Bộ chọn có thể nhắm mục tiêu tên View trong XAML. Sử dụng ký hiệu #ViewName để xác định quy tắc CSS cho các điều khiển mục nhập và trình chỉnh sửa có tên Themes và phản hồi bằng cách sử dụng Color, thuộc tính Font Stylesvà Color nền để chuyển phông chữ của Themes và phản hồi View nghiêng bằng Color Hải quân trên nền trắng, như được hiển thị trong danh sách 4-39.

***Liệt kê 4-39.*** Quy tắc CSS cho lượt View Themes có tên và phản hồi

#Subject,#Feedback{ color: navy; font-style: italic; background-color: white;

}

Bạn có thể cung cấp một đường viền cho dạng View phản hồi bằng cách xác định một kiểu Feedbackborder với một padding của 1 và Color nền bạc, như được hiển thị trong danh sách 4-40.

***Liệt kê 4-40.*** Quy tắc CSS để xác định một đường viền cho StackLayout chứa giao diện trình chỉnh sửa

.FeedbackBorder {

padding: 1; background-color: silver;

}

Xác định kiểu nhắm mục tiêu tất cả các dạng View cùng loại bằng cách sử dụng ký hiệuclassName . Để áp dụng Color nền Color xanh lá cây cho tất cả các lượt View contentpage và Color xám cho tất cả các chế View Grid và khoảng đệm 30 điểm ảnh cho cả hai loại dạng View, hãy sử dụng Color nền và padding thuộc tính, như đã chứng minh trong danh sách 4-41.

***Liệt kê 4-41.*** CSS Rule cho contentpages và Grid

AContentPage {

background-color: #4CAF50;

padding: 30 30;

}

AGrid {

background-color: #AAAAAA; padding: 30 30;

}

Tạo một tệp XAML mới được gọi là Csspage. XAML và sao chép stylespage. XAML vào đó. Thay thế mystyles bằng cách tải CSS làm Resources Dictionaries sử dụng cú pháp stylesheet, như được hiển thị trong danh sách 4-42.

***Liệt kê 4-42.*** Tải một CSS vào XAML

<ContentPage Title="Cascading Style Sheet" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" xmlns:local="clr- namespace:CSSExample" x:Class="CSSExample.CSSPage">

<ContentPage.Resources>

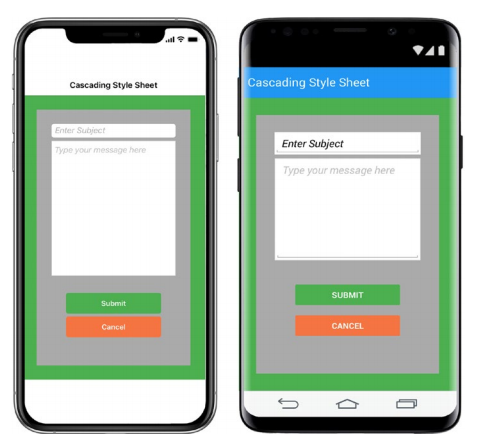
<StyleSheet Source="Styles/Styles.css" />

</ContentPage.Resources>

</ContentPage>

Loại bỏ các thiết lập thuộc tính Style cho stacklayout và hai Button và thay thế chúng bằng styleclass thuộc tính trỏ đến feedbackborder, submitbutton, và Button hủy bỏ như đã nêu trong Styles. CSS.

Hình 4-16 Hiển thị trang phản hồi bằng cách sử dụng CSS.



***Hình 4-16.*** *Trang phản hồi sử dụng CSS*

**Code đầy đủ: sử dụng CSS**

Danh sách 4-43 và danh sách 4-44 cung cấp Code hoàn chỉnh cho Styles. CSS và csspage. XAML.

***Liệt kê 4-43.*** Styles. CSS

^ContentPage { background-color: #4CAF50; padding: 30 30;

}

#Subject,#Feedback{ color: navy; font-style: italic; background-color: white;

}

.FeedbackBorder { padding: 1; background-color: silver;

}

.SubmitButton {

color: white; font-style: bold; background-color: #4CAF50;

}

.CancelButton { color: white; font-style: bold; background-color: #f47442;

}

.InactiveButton {

color: white; font-style: bold; background-color: gray;

}

^Grid {

background-color: #AAAAAA; padding: 30 30;

}

***Liệt kê 4-44.*** Csspage. XAML

<ContentPage Title="Cascading Style Sheet" xmlns="<http://xamarin.com/schemas/2014/forms>" xmlns:x="<http://schemas.microsoft.com/winfx/2009/xaml>" xmlns:local="clr- namespace:CSSExample" x:Class="CSSExample.CSSPage">

<ContentPage.Resources>

<StyleSheet Source="Styles/Styles.css" />

</ContentPage.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Entry Placeholder="Enter Subject" x:Name="Subject" Grid.Row="0" Focused="HandleSubject" Unfocused="HandleSubject" /> <StackLayout StyleClass="FeedbackBorder" Grid.Row="1" Orientation="Vertical">

<Editor x:Name="Feedback" Focused="HandleFeedback" Unfocused="HandleFeedback" VerticalOptions="FillAndExpand" /> </StackLayout>

<StackLayout Orientation="Vertical" Grid.Row="2">

<Button Text="Submit" StyleClass="SubmitButton" />

<Button Text="Cancel" StyleClass="CancelButton" /> </StackLayout>

</Grid>

</ContentPage>

**Xamarin. Forms định nghĩa CSS**

Sau đây là một đá Rosetta của CSS, dịch giữa XAML và CSS cho Selectors hỗ trợ. Một lần nữa, bất cứ điều gì bạn có thể làm trong CSS, bạn có thể làm trong XAML, và cách khác xung quanh. Bảng 4-1 và 4-2 danh sách hỗ trợ CSS Selectors, thuộc tính và Values Property trong xamarin. Forms.

***4-1.*** *Hỗ trợ CSS Selectors*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểm soát** | **Ví dụ CSS** | **XAML tương ứng** | **Mô tả** |
| . Class | . Note | < Label StyleClass = "ghi chú" > | Chọn lớp. Chọn bất kỳ Element nào phân loại là Note |
| #id | Message | < Label x:Name = "Submit" > | Bộ chọn ID. Chọn Element có tên duy nhất Message |
| \* | \* | - | Bộ chọn phổ biến. Chọn bất kỳ Element hình ảnh nào |
| Element | Label | < Label > | Bộ chọn Element. Chọn tất cả các Element hình ảnh của một loại cụ thể |
| Element Element | Label StackLayout | < StackLayoutxLabel/>  </StackLayout > | Chọn hậu duệ . Chọn tất cả Label bên trong StackLayout |
| Element> Element | StackLayout > Label | < StackLayoutxLabel/>  </StackLayout > | Chọn con. Chọn tất cả các Label trong đó cha mẹ là một StackLayout |
| Element + Element | Button + Label | < Label/xButton/> | Bên cạnh sibling Selector. Chọn tất cả các Button ngay lập tức được đặt sau Label |
| Element ~ Element | Label + Label | < Label/xButton/> | Anh chị Selector. Chọn tất cả các Button làm theo Label có cùng cha mẹ |
| Nguyên tố, nguyên tố | Label, Button | < Label >  Button < > | Danh sách các Selectors. Chọn tất cả Label và  Element Button |
| "Element | "VisualElement |  | Chọn loại cơ sở. Chọn bất kỳ Element nào dựa trên VisualElement. Bộ chọn này chỉ có sẵn trong Xamarin. Forms |

***Bảng 4-2.*** *Tính và Values được hỗ trợ bởi Xamarin. Forms*

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Visual Element** | **Value** | **Ví dụ** |
| background-color | VisualElement | Color type: Name, HEX, RGB, RGBA, HSL and HSLA values | background-color: red; |
| background-image | Page | string | background-image:logo.png; |
| border-color | Button, Frame | Color type | border-color:#222222; |
| border-width | Button | double | border-width:.25; |
| color | Button, DatePicker, Editor, Entry, Label, Picker, SearchBar, TimePicker | Color type | color:rgba(0,125,3,0.2); |
| direction | VisualElement | ltr, rtl,initial | direction:rtl; |
| font-family | Button, DatePicker, Editor,  Entry, Label, Picker, SearchBar, TimePicker, Span | string | font-family:Arial; |
| font-size | Button, DatePicker, Editor,  Entry, Label, Picker, SearchBar, TimePicker, Span | double  NamedSize type: default, micro, small, medium, large | font-size:Large; |
| font-style | Button, DatePicker, Editor,  Entry, Label, Picker, SearchBar, TimePicker, Span | bold, italic | font-style:italic; |
| height min-height width min-width | VisualElement | double | height:50; |
| margin margin-left margin-top margin-right margin-button | View | Thickness type double | margin:10 20; margin-right:3; |
| padding padding-left padding-top padding-right padding-bottom | Layout, Page | Thickness type double | padding: 10 20; padding-bottom:5; |
| opacity | VisualElement | double | opacity:.5; |
| text-align | Entry, EntryCell, Label, SearchBar | left, right, center, start, end | text-align:right; |
| visibility | VisualElement | true, visible, false, hidden, collapse | visibility: false; |

Tất cả các thuộc tính chấp nhận ban đầu là Values để đặt lại Values được đặt theo kiểu khác, ví dụ: Color nền: ban đầu;

Loại dữ liệu độ dày cho phép một (đồng nhất), hai (dọc ngang), ba (trên,  
ngang, dưới), và bốn (trên, bên phải, dưới, trái) Values được phân tách bằng khoảng trắng.

Tóm tắt

Sử dụng Resources trong Resources Dictionaries là một cách mạnh mẽ để giảm XAML trùng lặp và cải thiện khả năng duy trì của ứng dụng. Bất kỳ đối tượng nào cũng có thể được định nghĩa là Resources bên trong Dictionaries. Resources được View xét bằng phím hoặc một lần khi các yếu tố XAML sử dụng Resources được khởi tạo lần đầu tiên sử dụng phần mở rộng đánh dấu staticresource hoặc liên tục sử dụng dynamicresource. Hành vi tìm kiếm Resources cho phép Resources được xác định ở mức thấp hơn để ghi đè lên những xác định cao trong ứng dụng. Resources có thể được lưu trữ trong Dictionaries bên ngoài để tái sử dụng trên nhiều trang hoặc toàn bộ ứng dụng. Style Resources địa chỉ các hạn chế của việc sử dụng Resources trực tiếp để xác định giao diện người dùng thiết kế. Điều này cho phép tạo các ứng dụng performant, có thể đọc và mạch lạc hơn. Style có thể được định nghĩa ngầm hoặc Explicit, có thể được ghi đè tại địa phương, và hỗ trợ cả thừa kế Dynamic và Static. Xamarin. Các hình thức cung cấp thiết bị được Constructor trong Stylcho nhãn để đáp ứng các thiết lập khả năng tiếp cận của người dùng và đi kèm với hai Themes được xác định trước để hỗ trợ một tối và chế độ Light. Ngoài ra, để các cú pháp XAML, CSS ký hiệu có thể được sử dụng, mà tại thời gian chạy được dịch sang Style trong một Dictionaries resource.

Đó là Xamarin. Forms thiết kế sử dụng Resources, Style, và CSS. Hãy chuyển sang tương tác người dùng!