Xamarin XAML 语言教程内容呈现篇

(内部资料 v1.0)



大学霸 www.daxueba.net

前言

Xamarin 是一个跨平台开发框架。它可以用来开发 iOS、Android、Windows Phone 和 Mac 的应用程序。使用 Xamarin 框架中的 Forms 子框架,用户可以一次性的开发多个平台的应用,如 iOS、Android、Windows Phone,从而节省大量的开发时间。

在 Xamarin.Forms 中,用户可以直接使用 XAML 语言直接进行界面设计。这样,就可以将界面和逻辑代码分离,使得应用程序的结构更加清晰。为了满足大家的开发需求,本教程着眼于 Xamarin.Forms 开发,详细讲解 XAML 语言在界面设计中的使用。同时为了方便大家理解,我们为每个知识点都配以小实例。

1.学习所需的系统和软件

- □ 安装 Windows 10 操作系统
- ☐ Xamarin 4.2.0.719
- □ 安装 OS X 10.11
- □ 安装 Xcode 8.0

2.学习建议

大家学习之前,可以到百度网盘(下载链接: xxxxxxxxxxxxxxxx)获取相关的资料和软件。如果大家在学习过程遇到问题,也可以将问题发送到邮箱 xxxxxxxxxxxxxxx 。我们尽可能给大家解决。

目 录

第8章	文本 Label	1
8.1	标签控件的基本结构	1
	8.1.1 构建标签控件	1
	8.1.2 显示文本	1
	8.1.3 多行问题	2
	8.1.4 特殊字符	3
8.2	字体	
	8.2.1 显示文本的字体	4
	8.2.2 字体样式	5
	8.2.3 字体大小	<i>6</i>
	8.2.4 字体系列	11
8.3	颜色	
	8.3.1 文本颜色	
	8.3.2 背景颜色	
8.4	对齐	
	8.4.1 文本对齐	19
	8.4.2 缩进	23
	8.4.3 单行截断	
	8.4.4 换行	25
8.5	格式化文本	28
第9章	图像 Image	31
9.1	支持的属性和事件	31
	9.1.1 Image 类支持的 XAML	
	9.1.2 公共属性和事件	31
9.2	图像来源	32
	9.2.1 网络图像	32
	9.2.2 本地图像	33
9.3	图像缩放	37
	9.3.1 缩放模式	37
	9.3.2 缩放因子	40
9.4	图像定位	41
	9.4.1 描点位置	41
	9.4.2 布局方式	43
9.5	图像尺寸	45
	9.5.1 高度宽度	45
	9.5.2 最小的高度宽度	46

9.6	旋转图像	47
	9.6.1 Z轴上的旋转	47
	9.6.2 X 轴上的旋转	48
	9.6.3 Y 轴上的旋转	49
9.7	图像背景	50
	9.7.1 背景颜色	50
	9.7.2 透明度	51
9.8	图像尺寸改变	51
9.9		
	9.9.1 图像是否可见	
	9.9.2 图像是否加载	57
	9.9.3 图像是否启用	59
	9.9.4 图像是否具有焦点	61
9.10) 平台指定图像	
	9.10.1 使用 OnPlatform 标签	
	9.10.2 使用本地文件	
9.11	1 颜色块 BoxView	
	9.11.1 颜色块的颜色	
	9.11.2 颜色块的大小	
第 10 章		
10.1	1 点击操作 Button	
	10.1.1 构建按钮	
	10.1.2 按钮文本	
	10.1.3 背景颜色	
	10.1.4 边框	71
	10.1.5 图片	74
	10.1.6 事件触发	
10.2	2 微调操作	80
	10.2.1 滑块控件 Slider	
	10.2.2 步进控件 Stepper	
10.3	3 开关操作	
	10.3.1 开关控件 Switch	
	10.3.2 自定义复选框 CheckBox	
10.4	4 文本输入	
	10.4.1 键盘	
	10.4.2 单行文本——文本框控件 Entry	
	10.4.3 多行文本——文本视图 Editor	
	10.4.4 搜索栏	
第 11 章		
11.1	I 通用选择器 Picker	
	11.1.1 设置标题	
	11.1.2 设置条目	

1	1.1.3	实现响应	. 136
1	1.1.4	被选中的条目	. 139
1	1.1.5	选择器的数据绑定	. 140
11.2	日期选	E择器 DatePicker	. 143
1	1.2.1	设置日期	. 143
1	1.2.2	日期格式	. 146
1	1.2.3	实现响应	. 147
11.3	时间选	译器 TimePicker	. 150
1	1.3.1	设置显示的时间	. 150
1	1.3.2	设置时间格式	. 152
1	1.3.3	实现响应	. 154
		态与进度	
12.1	活动指	行示器 ActivityIndicator	. 157
	2.1.1	构建活动指示器	. 157
1:	2.1.2	活动指示器的显示隐藏	. 157
1:	2.1.3	活动指示器颜色	. 161
12.2	进度条	rogressBar	. 162
1:	2.2.1	构建进度条	. 162
1:	2.2.2	设置进度条的当前进度	. 163
1:	2.2.3	进度条的显示方式	. 169
第 13 章		容	
13.1	网页初	图图 WebView	. 173
1	3.1.1	加载内容	. 173
1		导航	
1	3.1.3	加载事件	. 178
1	3.1.4	注意问题	. 181
13.2	地图N	Лар	. 182
1	3.2.1	初始化地图	. 182
1:	3.2.2	指定显示内容	. 191
1	3.2.3	显示方式	. 197
13.3	OpenC	GL 视图 OpenGLView	. 199

第12章 程序状态与进度

在 Xamarin.Forms 中,提供了两个控件用来指示程序的状态和进度。他们分别为活动指示器和进度条。其中,活动指示器在程序正在等待长时间操作完成时使用,即在不确定进度的操作中使用,如加载网络视频,网页、图像等,而进度条是在确定进度时使用的,如下载图像、视频等。本章将讲解这两个控件。

12.1 活动指示器 ActivityIndicator

ActivityIndicator 被称为活动指示器,它给用户提供简单的反馈,表明程序正在运行,不提供具体进度信息。本节将讲解如何使用活动指示器。

12.1.1 构建活动指示器

要在 XAML 中构建活动指示器,就需要使用到 ActivityIndicator 标签,其语法如下:

<ActivityIndicator />

或者是:

<ActivityIndicator>

</ActivityIndicator>

12.1.2 活动指示器的显示隐藏

在构建好活动指示器后,它不会向滑块控件、步进控件、文本框控件等一样直接显示在界面中,这是因为此时的活动指示器是隐藏的。如果想要控制指示器的显示和隐藏,需要使用 ActivityIndicator 定义的 IsRunning 属性,此属性可以用来指示活动指示器是否正在运行,其语法形式如下:

<a href="mailto:

其中,boolValue 是一个布尔类型的值,当此值为 true 时,活动指示器会自动显示,并且运行;当此值为 false 时,活动指示器会自动隐藏。

【示例 12-1: ActivityIndicatorIsRunningOne】以下将活动指示器进行显示。代码如下:

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

<ContentPage xmIns="http://xamarin.com/schemas/2014/forms"</p>

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-namespace:ActivityIndicatorIsRunningOne" x:Class="ActivityIndicatorIsRunningOne.MainPage">

<ActivityIndicator IsRunning="True"

VerticalOptions="Center" />

</ContentPage>

此时运行程序,会看到如图 12.1~12.3 所示的效果







图 12.1 Android 的运行效果

图 12.2 iOS 的运行效果

图 12.3 Windows Phone 的运行效果

在此图中可以看到活动指示器在各个平台下的显示是有区别的。在 Android 中,活动指示器是一个 圆环状;在 iOS 中,活动指示器是一个锯齿状,就是常说的菊花;在 Windows Phone 中,活动指示器是 有 5 个小圆点组成的。

开发者除了可以在 XAML 中使用 IsRunning 属性控制指示器的显示隐藏外, 还可以在代码隐藏文件 中使用 IsRunning 属性控制指示器的显示隐藏。这时,首先需要在 XAML 文件中,使用 x:Name 属性为 活动指示器定义一个名称,然后在代码隐藏文件中通过定义的名称对 IsRunning 属性进行设置即可。

【示例 12-2: ActivityIndicatorIsRunningTwo】以下将在代码隐藏文件中控制活动指示器的显示和隐 藏。具体的操作步骤如下:

```
(1) MainPage.xaml 文件,编写代码,对内容页面进行布局。代码如下:
<?xml version="1.0" encoding="utf-8" ?>
```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"</p> xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-namespace:ActivityIndicatorIsRunningTwo" x:Class="ActivityIndicatorIsRunningTwo.MainPage">

<StackLayout Spacing="55"

VerticalOptions="Center">

<ActivityIndicator x:Name="activityIndicator" />

<StackLayout Spacing="10"> <Button x:Name="showButton"

> Text="ShowActivityIndicator" Clicked="OnShowActivityIndicator"/>

<Button x:Name="hideButton"

IsEnabled="False"

Text="HideActivityIndicator"

Clicked="OnHideActivityIndicator"/>

</StackLayout>

</StackLayout>

</ContentPage>

(2) 打开 MainPage.xaml.cs 文件,编写代码,实现通过按钮控制活动指示器的显示和隐藏功能。 代码如下:

```
using System;
using System.Collections.Generic;
using System.Ling;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
namespace ActivityIndicatorIsRunningTwo
    public partial class MainPage: ContentPage
    {
        public MainPage()
```

```
{
            InitializeComponent();
        }
        //显示活动指示器
        void OnShowActivityIndicator(object sender, EventArgs args)
        {
            activityIndicator.lsRunning = true;
            showButton.lsEnabled = false;
            hideButton.lsEnabled = true;
        }
        //隐藏活动指示器
        void OnHideActivityIndicator(object sender, EventArgs args)
        {
            activityIndicator.IsRunning = false;
            showButton.lsEnabled = true;
            hideButton.IsEnabled = false;
        }
    }
}
```

此时运行程序,会看到如图 12.4~12.6 所示的效果。当开发者轻拍 ShowActivityIndicator 按钮,会看到活动指示器显示了,并且进行活动,效果类似于图 12.7~12.9 所示。







图 12.4 Android 的运行效果

图 12.5 iOS 的运行效果

图 12.6 Windows Phone 的运行效果



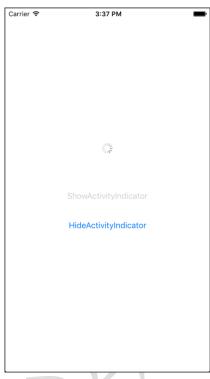




图 12.7 Android 的运行效果

图 12.8 iOS 的运行效果

图 12.9 Windows Phone 的运行效果

开发者除了可以为 ActivityIndicator 定义的 IsRunning 属性直接赋布尔值外,还可以通过数据绑定的方式为该属性赋值,此时绑定的数据也一样是返回布尔类型的。

【示例 12-3: ActivityIndicatorIsRunningThree】以下将实现加载网络图像的功能。代码如下:

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:ActivityIndicatorIsRunningThree"

x:Class="ActivityIndicatorIsRunningThree.MainPage">

<StackLayout>

<lmage x:Name="mylmage"</pre>

 $Source="https://timgsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1487155536489&di=cc531038dafb3b75d689fdd47a5a775c&imgtype=0&src=http%3A%2F%2Fpic1.win4000.com%2Fwallpaper%2F1%2F57973400dbbf3.jpg"$

HeightRequest="480"

WidthRequest="300"

HorizontalOptions="Center"/>

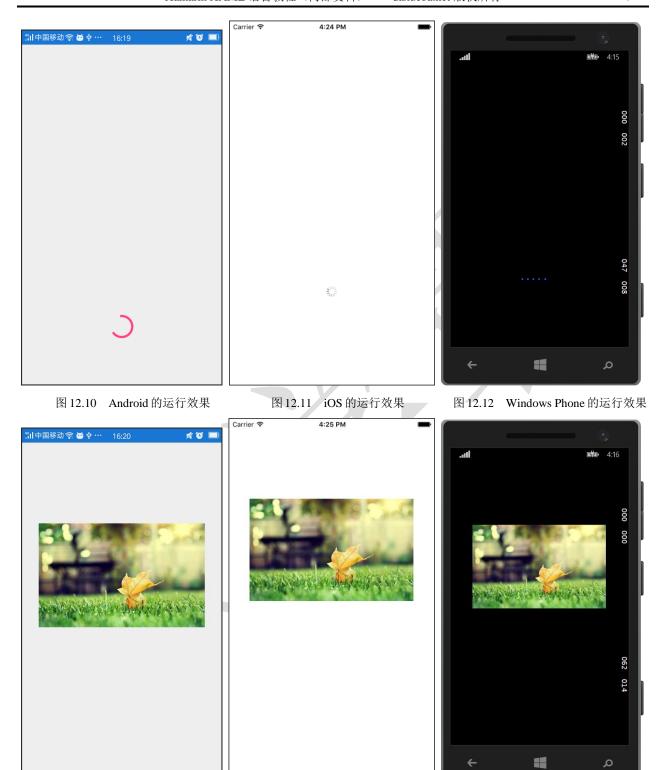
<ActivityIndicator x:Name="indicator"

BindingContext="{x:Reference Name=mylmage}" IsRunning="{Binding Path=IsLoading}"/>

</StackLayout>

</ContentPage>

在此代码中,我们将 ActivityIndicator 属性绑定到了 Image 的 IsLoading 属性上。此时运行程序,会看到如图 12.10~12.12 所示的效果。当图像加载完成后,会看到类似于图 12.13~12.15 所示的效果。



12.1.3 活动指示器颜色

图 12.13 Android 的运行效果

在图 12.10~12.12 中我们会看到在各个平台下活动指示器的颜色是不一样的。Android 的活动指示器

图 12.15 Windows Phone 的运行效果

图 12.14 iOS 的运行效果

默认是深粉色的; iOS 的活动指示器是灰色的; Windows Phone 的活动指示器是蓝色的。如果开发者想要让各个平台下活动指示器的颜色统一,可以使用 ActivityIndicator 定义的 Color 属性,其语法形式如下:

<a href="mailto:

其中,activityIndicatorColor 用来指定活动指示器的颜色,此颜色可以是 Color 中的静态字段也可以是带有或不带有 alpha 通道的十六进制的颜色名称。

【示例 12-4: ActivityIndicatorIsRunningOne】以下将以项目 ActivityIndicatorIsRunningOne 为基础,将活动指示器的颜色设置为绿色。代码如下:

<ActivityIndicator IsRunning="True"

Color="Green"

VerticalOptions="Center" />

此时运行程序,会看到如图 12.16~12.18 所示的效果



图 12.16 Android 的运行效果

图 12.17 iOS 的运行效果

图 12.18 Windows Phone 的运行效果

注意: Color 属性在 Windows Phone 中是无效的。

12.2 进度条 ProgressBar

ProgressBar 被称为进度条,它类似于没有滑块的滑块控件。进度条总是水平放置的。本节将讲解如何使用进度条。

注意: 进度条在各个平台下基本相同, 所以在后面的示例中我们只显示 Android 和 iOS 的运行效果。

12.2.1 构建进度条

要在 XAML 中构建进度条,就需要使用到 ProgressBar 标签,其语法如下:

<ProgressBar />

或者是:

<ProgressBar >

</ProgressBar >

【示例 12-5: CreateProgressBar】以下将在页面中构建一个进度条。代码如下:

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"</p>

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-namespace:CreateProgressBar"

x:Class="CreateProgressBar.MainPage">

<ProgressBar VerticalOptions="Center" />

</ContentPage>

此时运行程序,会看到如图 12.19~12.21 所示的效果。

图 12.19 Android 的运行效果

图 12.20 iOS 的运行效果

图 12.21 Windows Phone 的运行效果

12.2.2 设置进度条的当前进度

在图 12.19~12.21 中我们看到的是没有实现加载的进度条,即进度条的当前进度为 0,如果开发者想要修改当前进度,可以使用两种方式:一种是使用属性,另一种是使用方法。以下将讲解这两种方式。

1.使用属性

在 ProgressBar 中定义了一个 Progress 属性,此属性可以用来对进度条当前的进度进行设置。其语 法形式如下:

<ProgressBar Progress="progressValue" />

其中, progressValue 用来指定当前值,这个值是 Double 类型。

注意: ProgressBar 没有提供 MaxXX 和 MinXX 一类的相关属性, 所以 Progress 有效值是 0~1 范围内的 Double 类型数值。如果开发者将 Progress 的属性值设置为 0, 即表示进度条恢复初始状态。

【示例 12-6: CreateProgressBar】以下将以项目 CreateProgressBar 为基础,将进度条的当前进度设置为 0.8。代码如下:

<ProgressBar Progress="0.8"

VerticalOptions="Center" />

此时运行程序,会看到如图 12.22~12.23 所示的效果。



图 12.22 Android 的运行效果

图 12.23 iOS 的运行效果

开发者除了可以在 XAML 中使用 Progress 属性设置进度条的当前进度外,还可以在代码隐藏文件中使用 Progress 属性来设置进度条的当前进度。这时,首先需要在 XAML 文件中,使用 x:Name 属性为进度条定义一个名称,然后在代码隐藏文件中通过定义的名称对 Progress 属性进行设置即可。

【示例 12-7: ProgressBarProgressOne】以下将在代码隐藏文件中实现对进度条当前进行的设置。具体的操作步骤如下:

(1) MainPage.xaml 文件,编写代码,对内容页面进行布局。代码如下:

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"</p>

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-namespace:ProgressBarProgressOne" x:Class="ProgressBarProgressOne.MainPage">

<StackLayout Spacing="55"

VerticalOptions="Center">

<ProgressBar x:Name="progressBar" />

<StackLayout Spacing="10">

<Button Text="20%的进度"

Clicked="SetProgressPointTwo"/>

<Button Text="60%的进度"

Clicked="SetProgressPointSix"/>

<Button Text="100%的进度"

Clicked="SetProgressOne"/>

</StackLayout>

</StackLayout>

</ContentPage>

(2) 打开 MainPage.xaml.cs 文件,编写代码,实现通过按钮控制进度条当前进度的功能。代码如下:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
namespace ProgressBarProgressOne
   public partial class MainPage: ContentPage
       public MainPage()
       {
           InitializeComponent();
       //将进度条当前的进度设置为 0.2
       void SetProgressPointTwo(object sender, EventArgs args)
           progressBar.Progress = 0.2;
       }
       //将进度条当前的进度设置为 0.6
       void SetProgressPointSix(object sender, EventArgs args)
           progressBar.Progress = 0.6;
       //将进度条当前的进度设置为1
        void SetProgressOne(object sender, EventArgs args)
           progressBar.Progress = 1;
       }
   }
```

此时运行程序,会看到如图 12.24~12.25 所示的效果。当开发者轻拍某一按钮后,会看到进度条中显示对应的进度,效果类似于图 12.26~12.27 所示。



□ 11:30 AM □ 20%的进度 60%的进度 100%的进度

图 12.24 Android 的运行效果



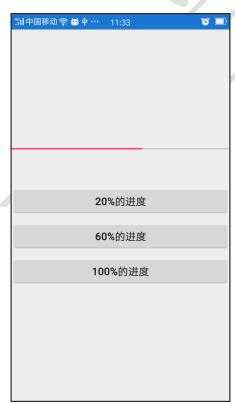


图 12.26 Android 的运行效果

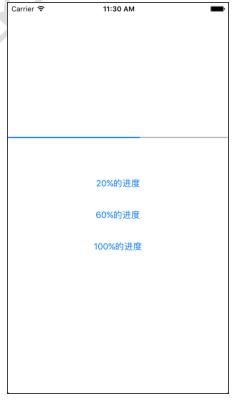


图 12.27 iOS 的运行效果

开发者除了可以为 ProgressBar 定义的 Progress 属性直接赋双精度类型的值外,还可以通过数据绑

定的方式为该属性赋值,此时绑定的数据也一样是返回双精度类型的。

【示例 12-8: ProgressBarProgressTwo】以下将通过滑块控件控制进度条的当前进度。代码如下:

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"</p>

xmlns: x = "http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:ProgressBarProgressTwo"

x:Class="ProgressBarProgressTwo.MainPage">

<StackLayout Spacing="60"

VerticalOptions="Center" >

<Slider x:Name="slider" />

<ProgressBar BindingContext="{x:Reference Name=slider}"
Progress="{Binding Path=Value}"/>

</StackLayout>

</ContentPage>

在此代码中, 我们将 ProgressBar 中的 Progress 属性绑定到了 Slider 的 Value 属性上。此时运行程序, 会看到如图 12.28~12.29 所示的效果。当开发者拖动滑块控件中的滑块时, 会看到类似于图 12.30~12.31 所示的效果。

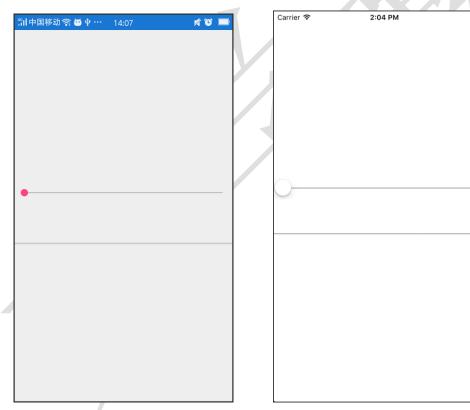
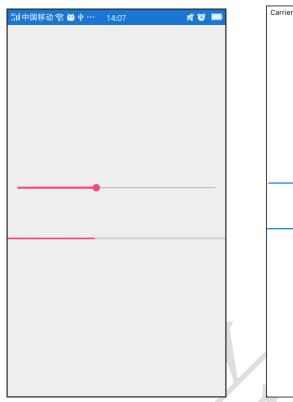


图 12.28 Android 的运行效果

图 12.29 iOS 的运行效果



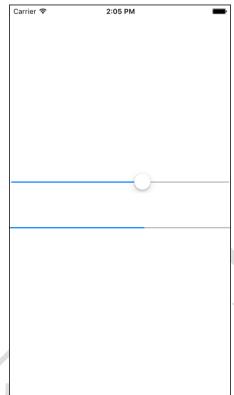


图 12.30 Android 的运行效果

图 12.31 iOS 的运行效果

2.使用方法

在 ProgressBar 中定义了一个 ProgressTo 方法,此方法也可以用来对进度条当前的进行进行设置,ProgressTo 与 Progress 属性的不同之处在于 ProgressTo 提供了动画效果。ProgressTo 方法必须要在代码文件中使用,不可以在 XAML 文件中使用。其语法形式如下:

ProgressBarObject. ProgressTo(value, length, easing);

其中,参数说明如下:

- □ value:表示设置的当前进度。
- □ length: 表示多少时间内达到设置的值(毫秒)。
- □ easing:表示动画效果。

【示例 12-9: ProgressBarProgressThree】以下将使用 ProgressTo 方法对进度条的当前进度进行设置。 具体的操作步骤如下:

- (1) MainPage.xaml 文件,编写代码,对内容页面进行布局。代码如下:
- <?xml version="1.0" encoding="utf-8" ?>
- <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:ProgressBarProgressThree"

x:Class="ProgressBarProgressThree.MainPage">

<StackLayout Spacing="55"

VerticalOptions="Center">

<ProgressBar x:Name="progressBar" />

<Button Text="PlayProgressAnimate"

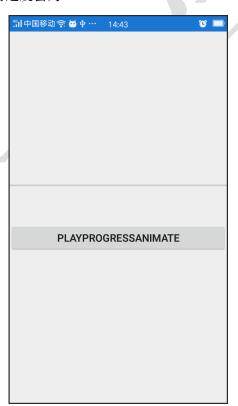
Clicked="SetProgresse"/>

</StackLayout>

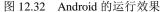
</ContentPage>

(2) 打开 MainPage.xaml.cs 文件,编写代码,对进度条的当前进度进行设置。代码如下: using System; using System.Collections.Generic; using System.Linq; using System.Text; using System.Threading.Tasks; using Xamarin.Forms; namespace ProgressBarProgressThree public partial class MainPage : ContentPage public MainPage() InitializeComponent(); //设置进度条的当前进度 void SetProgresse(object sender, EventArgs args) progressBar.ProgressTo(1.0, 2500, Easing.Linear); } } }

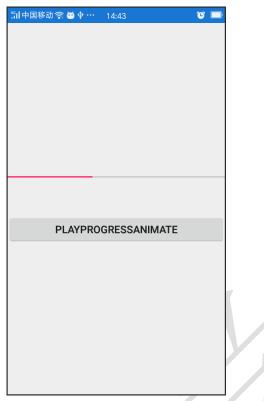
此时运行程序,会看到如图 12.32~12.33 所示的效果。当开发者轻拍 PlayProgressAnimate 按钮后,会看到进度条实现加载的动画效果,类似于图 12.34~12.35 所示的运行效果。2.5 秒后完成加载动画,进度条的当前进度会为 1。

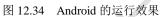












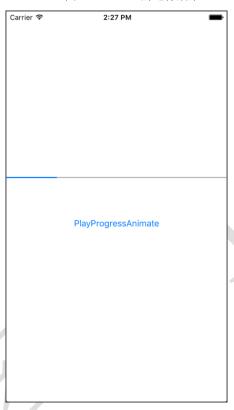


图 12.35 iOS 的运行效果

12.2.3 进度条的显示方式

在实际的应用开发中,进度条出现后,要求用户不能与应用程序有其它的交互。如果要实现此功能,就需要将进度条放置在一个单独的层中。触发此层出现,会看到进度条实现加载动画。当加载动画完成后,隐藏此层。一般称放置进度条的层称为遮罩层,此时的背景需要设置为半透明灰色。

【示例 12-10: ProgressBarDemo】以下将实现轻拍按钮后,全屏显示进度条的功能,并且进度条会实现加载动画。具体的操作步骤如下:

(1) MainPage.xaml 文件,编写代码,对内容页面进行布局。代码如下:

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

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"</p>

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-namespace:ProgressBarDemo"

x:Class="ProgressBarDemo.MainPage">

<AbsoluteLayout>

<StackLayout AbsoluteLayout.LayoutBounds="0, 0, 1, 1"

AbsoluteLayout.LayoutFlags="All">

<Image x:Name="myImage"</pre>

Source="http://benbenla.cn/images/20120721/benbenla-03c.jpg"

HeightRequest="480"

WidthRequest="300"

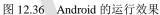
HorizontalOptions="Center"

```
VerticalOptions="Center"/>
          <Button Text="DownLoad"
                 Clicked="OnButtonClicked" />
        </StackLayout>
        <!-- 遮罩层 -->
        <ContentView x:Name="overlay"
                     AbsoluteLayout.LayoutBounds="0, 0, 1, 1"
                    AbsoluteLayout.LayoutFlags="All"
                    IsVisible="False"
                     BackgroundColor="#C0808080"
                    Padding="10, 0">
          <ProgressBar x:Name="progressBar"
                      VerticalOptions="Center" />
        </ContentView>
      </AbsoluteLayout>
    </ContentPage>
    在此代码中,我们将 Content View 元素设置为了遮罩层。在这个遮罩层中有一
层是不可见的。
     (2) 打开 MainPage.xaml.cs 文件,编写代码,实现遮罩层的显示隐藏、以及进度条的加载动画。
代码如下:
    using System;
    using System.Collections.Generic;
    using System.Ling;
    using System.Text;
    using System. Threading. Tasks;
    using Xamarin.Forms;
    namespace ProgressBarDemo
        public partial class MainPage : ContentPage
            public MainPage()
               InitializeComponent();
           void OnButtonClicked(object sender, EventArgs args)
               //显示遮罩层
               overlay.IsVisible = true;
               TimeSpan duration = TimeSpan.FromSeconds(5);
               DateTime startTime = DateTime.Now;
               //开启定时器
               Device.StartTimer(TimeSpan.FromSeconds(0.1), () =>
               {
                   double progress = (DateTime.Now - startTime).TotalMilliseconds /
                                     duration.TotalMilliseconds;
                   progressBar.Progress = progress;
                   bool continueTimer = progress < 1;
                   if (!continueTimer)
                   {
```

```
//隐藏遮罩
overlay.lsVisible = false;
}
return continueTimer;
});
}
}
```

此时运行程序,会看到如图 12.36~12.37 所示的效果。当开发者轻拍 DownLoad 按钮后,会看到具有进度条的遮罩层出现,不仅如此,进度条还会实现加载动画,类似于图 12.38~12.39 所示的效果。当进度条的当前进度为 1 时,具有进度条的遮罩层就会隐藏。





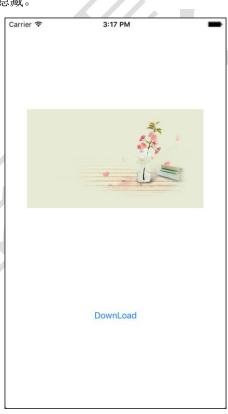


图 12.37 iOS 的运行效果



图 12.38 Android 的运行效果

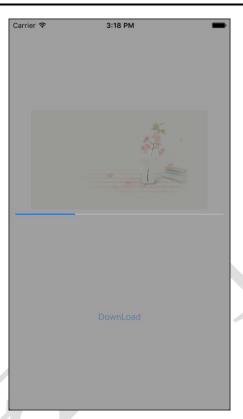


图 12.39 iOS 的运行效果