Gerando e lendo código de barras em Windows Phones

Este artigo explica como gerar e ler barcodes usando uma aplicação de Windows Phone e usando ZXing library.

Introdução

A barcode é uma forma de representar os dados de uma forma que pode ser lido por uma máquina. Todos os dias vemos os códigos de barras em todo o mundo ao nosso redor. Em cada embalagem de alimentos, cada livro, cada CD, vamos notar que cada item tem um código de barras, que é usado para identificar o item que está anexado. Há uma grande quantidade de formatos e tipos de códigos de barras. Alguns são simplesmente grupo de linhas verticais paralelas umas às outras. Através da variação da largura de cada linha, os dados podem ser codificados nos mesmos. Estes são chamados 1D (unidimensional) de códigos de barras. Outros codificam os dados utilizando um conjunto de padrões geométricos. Estes são chamados de códigos de barras 2D, ou códigos de matriz e são capazes de armazenar mais dados. Um exemplo comum de um código de barras 2D é um formato chamado Código QR. Com a revolução smartphone, QR Codes se tornou realmente popular. Em anúncios, por exemplo, podemos rapidamente ir para o site do produto, verificando a QR Code. Ou podemos partilhar as informações de contato usando um código QR.

Por omissão, o Windows Phone 7 e Windows Phone 8 são capazes de ler alguns tipos de códigos de barras (QR Code, EAN-13, Microsoft Tag), utilizando a aplicação Bing Vision (para usá-lo, pressione o botão de busca e selecione o ícone do olho no menu da aplicação). No entanto, por vezes, você precisa ler um código de barras que Bing Vision não suporta, ou que você vai querer ser capaz de adicionar funcionalidades de código de barras na sua aplicação (por exemplo, em uma aplicação para acompanhar o DVD de seus amigos emprestados). Bibliotecas como esta normalmente suportar uma grande quantidade de tipos de códigos de barras e pode ajudá-lo nessas tarefas.

Neste artigo vamos aprender como podemos adicionar a capacidade de gerar e digitalizar códigos de barras usando a câmara do telefone para a sua aplicação de Windows Phone. Neste exemplo, vamos criar uma aplicação simples que gera um QR Code contendo um número de telefone que você deseja partilhar e é capaz de ler códigos de barras e exibir os dados armazenados neles.

Bibliotecas

Vamos usar uma biblioteca chamada "ZXing.Net", que é um versão .Net do popular ZXing open-source projecto que oferece aos desenvolvedores a capacidade de ler e gerar códigos de barras. ZXing.Net está disponível como pacote NuGet, por isso vamos usar este método para adicionar essa referência para o nosso projeto.

Ir para o Solution Explorer, clique direito no ficheiro de projeto e selecione **Manage NuGet Packages**. Procure por "ZXing.Net" e clique em Instalar. Você também pode usar o Package Manager Console para instalar os pacotes. Execute o seguinte comando para instalar *ZXing.Net* usando este método: `PM> Install-Package ZXing.Net`.

Gerar barcodes

Neste exemplo vamos gerar um QR Code contendo um número de telefone. Vamos usar a classe PhoneNumberChooserTask para iniciar o aplicativo Contatos e para obter o número de telefone de um contato selecionado pelo utilizador.

```
PhoneNumberChooserTask phoneNumberChooserTask;
phoneNumberChooserTask = new PhoneNumberChooserTask();
phoneNumberChooserTask.Completed += phoneNumberChooserTask_Completed;
```

Agora vamos gerar o código QR com o número de telefone selecionado:

```
private static WriteableBitmap GenerateQRCode(string phoneNumber)
{
    BarcodeWriter _writer = new BarcodeWriter();

    _writer.Renderer = new ZXing.Rendering.WriteableBitmapRenderer()
    {
        Foreground = System.Windows.Media.Color.FromArgb(255, 0, 0, 255) // blue
```


(C) Copyright Nokia 2014. All rights reserved.

```
    };

    _writer.Format = BarcodeFormat.QR_CODE;


    _writer.Options.Height = 400;
    _writer.Options.Width = 400;
    _writer.Options.Margin = 1;

    var barcodeImage = _writer.Write("tel:" + phoneNumber); //tel: prefix for phone
numbers

    return barcodeImage;
}
```

O `BarcodeWriter` tem algumas propriedades que permitem que uma grande quantidade de personalização:

- A propriedade “Renderer” é usado para alterar a aparência do código de barras, como o fundo e as cores de primeiro plano, tamanho da fonte, o peso e a família. Neste exemplo vamos definir o azul como cor de primeiro plano, que irá criar um QR Code azul em um fundo branco.
- A propriedade “Formato” é usado para selecionar o formato de código de barras para gerar. Para este exemplo, vamos gerar um QR Code.
- A propriedade “Opções” permite especificar as dimensões do código de barras para gerar, bem como definir suas margens.

Irá notar que, neste caso, o prefixo dos dados codificados é `tel:`. Isto é para permitir que os leitores de códigos QR entendam que o conteúdo deste código de barras é um número de telefone. Este tipo particular de códigos de barras, o QR Code, suporta um monte de prefixos. Você pode encontrar uma lista detalhada destes padrões aqui.

Aqui está um exemplo de um QR Code gerado:

(C) Copyright Nokia 2014. All rights reserved.

A sample QR Code with a phone number

Estes são alguns exemplos de códigos de barras que podem ser gerados:

Aztec Code

EAN-13

(C) Copyright Nokia 2014. All rights reserved.

MSI PDF417

Ler barcodes

Adicionando recursos de digitalização de códigos de barras para a sua aplicação, por vezes, pode ser muito útil. Por exemplo, um aplicativo de pesquisa de preço seria muito mais fácil de usar se você poderia apenas apontar a câmara do seu telefone no código de barras e clicar em pesquisar.

No nosso exemplo, vamos adicionar o recurso de verificação em uma página diferente chamado “Scan.xaml”. Esta página contém um visor de modo que o utilizador pode ver o que a câmara está capturando e uma área para exibir informações sobre o código de barras. Partes deste exemplo foram retirados estes artigos: 28v = vs.105% 29.aspx How to create a base camera app for Windows Phone (*) e 28v = vs.105% 29.aspx How to use camera focus in an app for Windows Phone(*). (*) Artigo disponível apenas em inglês. Este é o código XAML para a página:

```
<!--LayoutRoot is the root grid where all page content is placed-->
<Grid x:Name="LayoutRoot" Background="Transparent">
    <Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="100" />
    </Grid.RowDefinitions>

    <Canvas x:Name="viewfinderCanvas">

        <!--Camera viewfinder -->
        <Canvas.Background>

            <VideoBrush x:Name="viewfinderBrush">
                <VideoBrush.RelativeTransform>
                    <CompositeTransform
                        x:Name="viewfinderTransform"
                        CenterX="0.5"
                        CenterY="0.5"
                        Rotation="90"/>
                </VideoBrush.RelativeTransform>
            </VideoBrush>
        </Canvas.Background>
        <TextBlock
            x:Name="focusBrackets"
```


(C) Copyright Nokia 2014. All rights reserved.

```
            Text="[   ]"
            FontSize="40"
            Visibility="Collapsed"/>
    </Canvas>
    <!--Used for debugging >-->
    <StackPanel Grid.Row="1" Margin="20, 0">
        <TextBlock x:Name="tbBarcodeType" FontWeight="ExtraBold" />
        <TextBlock x:Name="tbBarcodeData" FontWeight="ExtraBold" TextWrapping="Wrap" />
    </StackPanel>
</Grid>
```

This is the code behind:

```
public partial class Scan : PhoneApplicationPage
{
    private PhotoCamera _phoneCamera;
    private IBarcodeReader _barcodeReader;
    private DispatcherTimer _scanTimer;
    private WriteableBitmap _previewBuffer;

    public Scan()
    {
        InitializeComponent();
    }

    protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs
e)
    {
        // Initialize the camera object
        _phoneCamera = new PhotoCamera();
        _phoneCamera.Initialized += cam_Initialized;
        _phoneCamera.AutoFocusCompleted += _phoneCamera_AutoFocusCompleted;

        CameraButtons.ShutterKeyHalfPressed += CameraButtons_ShutterKeyHalfPressed;

        //Display the camera feed in the UI
        viewfinderBrush.SetSource(_phoneCamera);


        // This timer will be used to scan the camera buffer every 250ms and scan for
any barcodes
        _scanTimer = new DispatcherTimer();
        _scanTimer.Interval = TimeSpan.FromMilliseconds(250);
        _scanTimer.Tick += (o, arg) => ScanForBarcode();

        viewfinderCanvas.Tap += new EventHandler<GestureEventArgs>(focus_Tapped);

        base.OnNavigatedTo(e);
    }

    void _phoneCamera_AutoFocusCompleted(object sender,
CameraOperationCompletedEventArgs e)
    {
        Deployment.Current.Dispatcher.BeginInvoke(delegate()
        {
            focusBrackets.Visibility = Visibility.Collapsed;
```


(C) Copyright Nokia 2014. All rights reserved.

```
        });
    }

    void focus_Tapped(object sender, GestureEventArgs e)
    {
        if (_phoneCamera != null)
        {
            if (_phoneCamera.IsFocusAtPointSupported == true)
            {
                // Determine the location of the tap.
                Point tapLocation = e.GetPosition(viewfinderCanvas);

                // Position the focus brackets with the estimated offsets.
                focusBrackets.SetValue(Canvas.LeftProperty, tapLocation.X - 30);
                focusBrackets.SetValue(Canvas.TopProperty, tapLocation.Y - 28);

                // Determine the focus point.
                double focusXPercentage = tapLocation.X / viewfinderCanvas.ActualWidth;
                double focusYPercentage = tapLocation.Y / viewfinderCanvas.ActualHeight;

                // Show the focus brackets and focus at point.
                focusBrackets.Visibility = Visibility.Visible;
                _phoneCamera.FocusAtPoint(focusXPercentage, focusYPercentage);
            }
        }
    }

    void CameraButtons_ShutterKeyHalfPressed(object sender, EventArgs e)
    {
        _phoneCamera.Focus();
    }

    protected override void
OnNavigatingFrom(System.Windows.Navigation.NavigatingCancelEventArgs e)
    {
        //we're navigating away from this page, we won't be scanning any barcodes
        _scanTimer.Stop();

        if (_phoneCamera != null)
        {
            // Cleanup
            _phoneCamera.Dispose();
            _phoneCamera.Initialized -= cam_Initialized;
            CameraButtons.ShutterKeyHalfPressed -= CameraButtons_ShutterKeyHalfPressed;
        }
    }

    void cam_Initialized(object sender,
Microsoft.Devices.CameraOperationCompletedEventArgs e)
    {
        if (e.Succeeded)
        {
            this.Dispatcher.BeginInvoke(delegate()
            {
                _phoneCamera.FlashMode = FlashMode.Off;
                _previewBuffer = new
WriteableBitmap((int)_phoneCamera.PreviewResolution.Width,
```


(C) Copyright Nokia 2014. All rights reserved.

```
(int)_phoneCamera.PreviewResolution.Height);

                _barcodeReader = new BarcodeReader();

                // By default, BarcodeReader will scan every supported barcode type
                // If we want to limit the type of barcodes our app can read,
                // we can do it by adding each format to this list object

                //var supportedBarcodeFormats = new List<BarcodeFormat>();
                //supportedBarcodeFormats.Add(BarcodeFormat.QR_CODE);
                //supportedBarcodeFormats.Add(BarcodeFormat.DATA_MATRIX);
                //_bcReader.PossibleFormats = supportedBarcodeFormats;

                _barcodeReader.TryHarder = true;

                _barcodeReader.ResultFound += _bcReader_ResultFound;
                _scanTimer.Start();
            });
        }
        else
        {
            Dispatcher.BeginInvoke(() =>
                {
                    MessageBox.Show("Unable to initialize the camera");
                });
        }
    }

    void _bcReader_ResultFound(Result obj)
    {
        // If a new barcode is found, vibrate the device and display the barcode details
in the UI
        if (!obj.Text.Equals(tbBarcodeData.Text))
        {
            VibrateController.Default.Start(TimeSpan.FromMilliseconds(100));
            tbBarcodeType.Text = obj.BarcodeFormat.ToString();
            tbBarcodeData.Text = obj.Text;
        }
    }

    private void ScanForBarcode()
    {
        //grab a camera snapshot
        _phoneCamera.GetPreviewBufferArgb32(_previewBuffer.Pixels);
        _previewBuffer.Invalidate();

        //scan the captured snapshot for barcodes
        //if a barcode is found, the ResultFound event will fire
        _barcodeReader.Decode(_previewBuffer);

    }

}
```

Depois de ter uma aplicação de câmara em funcionamento, verificação de códigos de barras é realmente muito simples. A biblioteca ZXing.Net contém uma classe chamada `BarcodeReader`. O método `Decode` permite digitalizar um `WriteableBitmap` para

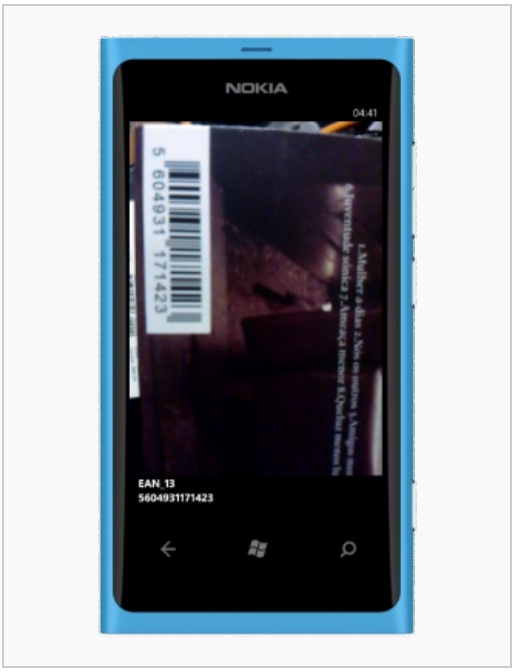
(C) Copyright Nokia 2014. All rights reserved.

códigos de barras. Se qualquer código de barras é encontrado, o `resultadoencontrado` evento dispara.

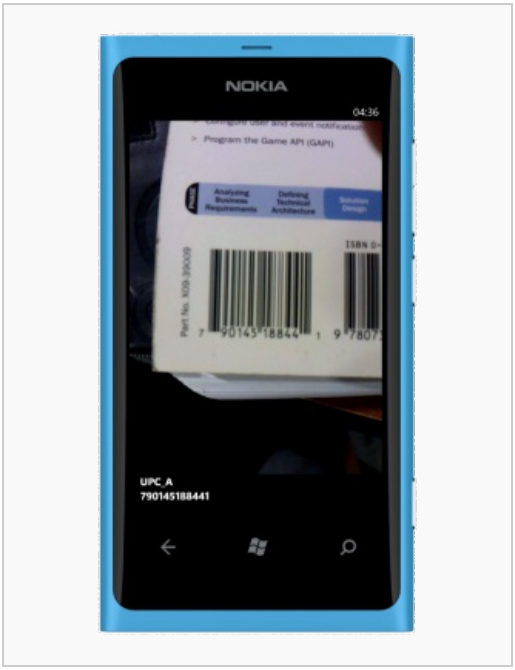
Neste exemplo, um `DispatcherTimer _scanTimer` é usada para obter o que a câmera está capturando e atribui-lo na classe `BarcodeReader`. Por omissão, a classe `BarcodeReader` fará o "scan" de todos os formatos de código de barras suportados em uma imagem. Se quiser restrigir este processo apenas a alguns formatos, tudo o que você precisa fazer é definir o `PossibleFormats` no `BarcodeReader` objeto. Se tiver problemas para digitalizar alguns códigos de barras, códigos especialmente 1D, poderá tentar definir a propriedade `TryHarder` igual a `True`. *Note-se que isso terá algum impacto sobre o desempenho, mas pelo menos no meu Lumia 800 não é tão percetível.*

Aqui estão alguns exemplos de códigos de barras digitalizados:

Aztec Code



EAN-13



UPC

Sumário

Neste artigo abordamos a digitalização e geração de códigos de barras na plataforma Windows Phone usando a biblioteca ZXing.Net. Ele suporta uma ampla gama de plataformas, o que pode ser útil se não estiver focado só em Windows Phone. Como se trata de uma biblioteca com origem numa biblioteca de aplicações para Android, é possível encontrar imensa documentação que poderá ajudar. Sinta-se livre para obter a aplicação de exemplo (File:WPBarcodes.zip) e brincar com ele!

(C) Copyright Nokia 2014. All rights reserved.

Existem outras bibliotecas que oferecem a mesma funcionalidade, por exemplo, os Telerik RadControls para Windows Phone. Se você é membro da Nokia Developer Program premium para Lumia, você pode acede-los gratuitamente.

(C) Copyright Nokia 2014. All rights reserved.