Kostenloses eBook

LERNEN

Xamarin.Forms

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.fo

rms

Inhaltsverzeichnis

Über ... 1

Kapitel 1: Erste Schritte mit Xamarin.Forms ... 2

Bemerkungen ... 2

Versionen ... 2

Examples ... 3

Installation (Visual Studio) ... 3

Xamarin Plugin für Visual Studio ... 3

Xamarin.Forms ... 4

Hello World Xamarin Forms: Visual Studio ... 5

Schritt 1: Ein neues Projekt erstellen. ... 5

Schritt 2: Untersuchung der Probe ... 6

Schritt 3: Starten der Anwendung ... 7

Kapitel 2: Abhängigkeitsdienste ... 8

Bemerkungen ... 8

Examples ... 8

Greifen Sie auf Kamera und Galerie zu. ... 8

Kapitel 3: Alert anzeigen ... 9

Examples ... 9

DisplayAlert ... 9

Alert-Beispiel mit nur einer Taste und Aktion ... 10

Kapitel 4: AppSettings Reader in Xamarin.Forms ... 11

Examples ... 11

Einlesen der app.config-Datei in einem Xamarin.Forms-Xaml-Projekt ... 11

Kapitel 5: Auslöser und Verhalten ... 13

Examples ... 13

Xamarin Forms Trigger Beispiel ... 13

Multi-Trigger ... 14

Kapitel 6: Ausnahmebehandlung ... 16

Examples ... 16

Eine Möglichkeit, über iOS Ausnahmen zu berichten ... 16
Kapitel 7: Auswirkungen ... 19
Einführung ... 19
Examples ... 19
Plattformspezifischen Effekt für ein Entry-Control hinzufügen ... 19
Kapitel 8: BDD-Unit-Tests in Xamarin.Forms ... 24
Bemerkungen ... 24
Examples ... 24
Simple Specflow zum Testen von Befehlen und der Navigation mit NUnit Test Runner ... 24
Warum brauchen wir das? ... 24
Verwendungszweck: ... 24
Erweiterte Verwendung für MVVM ... 26
Kapitel 9: Benutzerdefinierte Renderer ... 28
Examples ... 28
Benutzerdefinierter Renderer für ListView ... 28
Benutzerdefinierter Renderer für BoxView ... 30
Auf den Renderer von einem nativen Projekt aus zugreifen ... 34
Abgerundetes Label mit einem benutzerdefinierten Renderer für Frame (PCL- und iOS-Teile) ... 34
Abgerundete BoxView mit wählbarer Hintergrundfarbe ... 35
Kapitel 10: Benutzerdefinierte Schriftarten in Stilen ... 38
Bemerkungen ... 38
Examples ... 38
Zugriff auf benutzerdefinierte Schriftarten in Syles ... 38
Kapitel 11: Benutzerdefinierte Steuerelemente erstellen ... 41
Examples ... 41
Erstellen Sie ein benutzerdefiniertes Steuerelement für Xamarin Forms (keine native Eingab ... 41
Beschriftung mit bindbarer Sammlung von Feldern ... 44
Erstellen eines benutzerdefinierten Eintragssteuerelements mit einer MaxLength-Eigenschaft ... 45
Kapitel 12: Benutzerdefinierte Steuerelemente erstellen ... 47
Einführung ... 47
Examples ... 47

Ein CheckBox-Steuerelement implementieren ... 47
Erstellen des benutzerdefinierten Steuerelements ... 47
Verbrauch der benutzerdefinierten Steuerung ... 48
Erstellen des benutzerdefinierten Renderers auf jeder Plattform ... 48
Erstellen des benutzerdefinierten Renderers für Android ... 49
Erstellen des benutzerdefinierten Renderers für iOS ... 50
Kapitel 13: Benutzerdefinierte Steuerelemente erstellen ... 55
Examples ... 55
Benutzerdefinierte Schaltfläche erstellen ... 55
Kapitel 14: Caching ... 57
Examples ... 57
Caching mit Akavache ... 57
Über Akavache ... 57
Empfehlungen für Xamarin ... 57
Einfaches Beispiel ... 57
Fehlerbehandlung ... 58
Kapitel 15: CarouselView - Vorabversion ... 59
Bemerkungen ... 59
Examples ... 59
Karussellansicht importieren ... 59
CarouselView in eine XAML-Seite importieren ... 60
Die Grundlagen ... 60
Bindbare Quelle erstellen ... 60
DataTemplates ... 60
Kapitel 16: Datenbindung ... 62
Bemerkungen ... 62
Mögliche Ausnahmen ... 62
System.ArrayTypeMismatchException: Es wurde versucht, auf ein Element als Typ zuzugreifen, ... 62
System.ArgumentException: Objekt vom Typ 'Xamarin.Forms.Binding' kann nicht in den Typ 'Sy ... 62
Die Picker.Items Eigenschaft ist nicht bindbar ... 62
Examples ... 63

Grundlegende Bindung an ViewModel ... 63
Kapitel 17: DependencyService ... 65
Bemerkungen ... 65
Examples ... 65
Schnittstelle ... 65
iOS-Implementierung ... 65
Geteilter Code ... 66
Android-Implementierung ... 67
Kapitel 18: Generischer Xamarin.Forms-App-Lebenszyklus? Plattformabhängig! ... 69
Examples ... 69
Der Lebenszyklus von Xamarin.Forms ist nicht der eigentliche App-Lebenszyklus, sondern ein ... 69
Kapitel 19: Gesten ... 71
Examples ... 71
Machen Sie ein Bild tappbar, indem Sie einen TapGestureRecognizer hinzufügen ... 71
Zoomen Sie ein Bild mit der Pinch-Geste ... 71
Zeigen Sie den gesamten vergrößerten Bildinhalt mit PanGestureRecognizer an ... 72
Platzieren Sie eine Nadel an der Stelle, an der der Benutzer den Bildschirm mit MR.Gesture ... 72
Kapitel 20: Kontaktauswahl - Xamarin-Formulare (Android und iOS) ... 74
Bemerkungen ... 74
Examples ... 74
contact_picker.cs ... 74
MyPage.cs ... 74
Wählen SieContactPicker.cs ... 75
Wählen SieContactActivity.cs ... 75
MainActivity.cs ... 77
Wählen SieContactRenderer.cs ... 77
Kapitel 21: ListViews verwenden ... 80
Einführung ... 80
Examples ... 80
Zum Aktualisieren in XAML und hinterem Code ziehen ... 80
Kapitel 22: MessagingCenter ... 81
Einführung ... 81

Examples ... 81
Einfaches Beispiel ... 81
Argumente übergeben ... 82
Abbestellen ... 82
Kapitel 23: Mit Karten arbeiten ... 84
Bemerkungen ... 84
Examples ... 84
Hinzufügen einer Karte in Xamarin.Forms (Xamarin Studio) ... 84
Karteninitialisierung ... 84
iOS-Projekt ... 84
Android-Projekt ... 84
Plattformkonfiguration ... 85
iOS-Projekt ... 85
Android-Projekt ... 86
Hinzufügen einer Karte ... 96
PCL-Projekt ... 96
Kapitel 24: Mit lokalen Datenbanken arbeiten ... 98
Examples ... 98
Verwenden von SQLite.NET in einem freigegebenen Projekt ... 98
Arbeiten mit lokalen Datenbanken mit xamarin.forms in Visual Studio 2015 ... 100
Kapitel 25: Mitteilungen ... 110
Bemerkungen ... 110
Examples ... 110
Push-Benachrichtigungen für iOS mit Azure ... 110
Push-Benachrichtigungen für Android mit Azure ... 113
Push-Benachrichtigungen für Windows Phone mit Azure ... 116
Kapitel 26: Mitteilungen ... 118
Bemerkungen ... 118
AWS Simple Benachrichtigungsdienst Lingo: ... 118
Generisches Pushbenachrichtigungs-Lingo: ... 118
Examples ... 118

iOS-Beispiel ... 118
Kapitel 27: Navigation in Xamarin.Forms ... 120
Examples ... 120
NavigationSeitenfluss ... 120
Navigationsseitenfluss mit XAML ... 121
Hierarchische Navigation mit XAML ... 122
Neue Seiten schieben ... 122
Page1.xaml ... 123
Page1.xaml.cs ... 123
Page2.xaml ... 123
Page2.xaml.cs ... 123
Seiten knallen ... 124
Page3.xaml ... 124
Page3.xaml.cs ... 124
Modale Navigation mit XAML ... 124
Vollbildmodale ... 125
Alarme / Bestätigungen und Benachrichtigungen ... 125
Aktionsblätter ... 125
Master-Detailseite ... 125
Master Detail Navigation ... 126
Kapitel 28: Navigation in Xamarin.Forms ... 127
Bemerkungen ... 127
Examples ... 127
Verwendung von INavigation aus dem Ansichtsmodell ... 127
Kapitel 29: OAuth2 ... 131
Examples ... 131
Authentifizierung mit dem Plugin ... 131
Kapitel 30: Plattformspezifische visuelle Anpassungen ... 133
Examples ... 133
Sprachanpassungen ... 133
Plattformanpassungen ... 133
Stile verwenden ... 134

Benutzerdefinierte Ansichten verwenden ... 134
Kapitel 31: Plattformspezifisches Verhalten ... 136
Bemerkungen ... 136
Examples ... 136
Symbol im Navigationsheader in Anroid entfernen ... 136
Verkleinern Sie die Schriftgröße des Labels in iOS ... 137
Kapitel 32: SQL-Datenbank und API in Xamarin-Formularen. ... 139
Bemerkungen ... 139
Examples ... 139
API mit SQL-Datenbank erstellen und in Xamarin-Formularen implementieren ... 139
Kapitel 33: Unit Testing ... 140
Examples ... 140
Testen der Ansichtsmodelle ... 140
Bevor wir anfangen... ... 140
Geschäftsanforderungen ... 140
Gemeinsame Klassen ... 141
Dienstleistungen ... 141
Erstellen des ViewModel-Stubs ... 142
Wie erstelle ich eine LoginPageViewModel-Instanz? ... 143
Tests ... 143
Tests schreiben ... 144
Implementierung der Geschäftslogik ... 145
Kapitel 34: Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet? ... 147
Bemerkungen ... 147
Examples ... 147
Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet? ... 147
Kapitel 35: Xamarin Plugin ... 149
Examples ... 149
Plugin freigeben ... 149
ExternalMaps ... 149
Geolocator Plugin ... 150

Medien Plugin ... 152
Messaging-Plugin ... 155
Plugins für Berechtigungen ... 157
Kapitel 36: Xamarin Relatives Layout ... 161
Bemerkungen ... 161
Examples ... 161
Seite mit einem einfachen Etikett in der Mitte ... 161
Kiste für Kiste ... 163
Kapitel 37: Xamarin.Forms Cells ... 166
Examples ... 166
EntryCell ... 166
SwitchCell ... 166
TextCell ... 167
ImageCell ... 168
ViewCell ... 169
Kapitel 38: Xamarin.Forms Seite ... 171
Examples ... 171
TabbedPage ... 171
Inhaltsseite ... 172
MasterDetailPage ... 173
Kapitel 39: Xamarin.Forms Views ... 175
Examples ... 175
Taste ... 175
Datumsauswahl ... 176
Eintrag ... 177
Editor ... 178
Bild ... 179
Etikette ... 180
Kapitel 40: Xamarin-Formularlayouts ... 182
Examples ... 182
ContentPresenter ... 182
ContentView ... 182

Rahmen ... 183
ScrollView ... 184
TemplatedView ... 186
AbsoluteLayout ... 186
Gitter ... 189
RelativesLayout ... 191
StackLayout ... 192
Verwendung in XAML ... 193
Verwendung im Code ... 193
Kapitel 41: Xamarin-Geste ... 196
Examples ... 196
Tippen Sie auf Geste ... 196
Kapitel 42: Xamarin-Geste ... 197
Examples ... 197
Gestenereignis ... 197
Kapitel 43: Zugriff auf native Funktionen mit DependencyService ... 200
Bemerkungen ... 200
Examples ... 200
Implementierung von Text-zu-Sprache ... 200
iOS-Implementierung ... 201
Android-Implementierung ... 202
Windows Phone-Implementierung ... 203
Implementierung in Shared Code ... 204
Versionsnummern von Anwendungs- und Geräte-Betriebssystemen erhalten - Android & iOS - PCL ... 204
Credits ... 207

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xamarin-forms

It is an unofficial and free Xamarin.Forms ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Xamarin.Forms.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Xamarin.Forms

Bemerkungen

Xamarin.Forms ermöglicht das Erstellen von iOS-, Android- und Windows-Apps mit einer großen Menge an gemeinsamem Code, einschließlich UI-Code oder XAML-UI-Markup. App-Seiten und -Ansichten sind auf jeder Plattform nativen Steuerelementen zugeordnet, können jedoch angepasst werden, um eine plattformspezifische Benutzeroberfläche bereitzustellen oder auf plattformspezifische Funktionen zuzugreifen.

Versionen

Ausführung	Veröffentlichungsdatum
2.3.1	2016-08-03
2.3.0-hotfix1	2016-06-29
2.3.0	2016-06-16
2.2.0-hotfix1	2016-05-30
2.2.0	2016-04-27
2.1.0	2016-03-13
2.0.1	2016-01-20
2.0.0	2015-11-17
1.5.1	2016-10-20
1.5.0	2016-09-25
1.4.4	2015-07-27
1.4.3	2015-06-30
1.4.2	2015-04-21
1.4.1	2015-03-30
1.4.0	2015-03-09
1.3.5	2015-03-02
1.3.4	2015-02-17

Ausführung	Veröffentlichungsdatum
1.3.3	09.02.2015
1.3.2	2015-02-03
1.3.1	2015-01-04
1.3.0	2014-12-24
1.2.3	2014-10-02
1.2.2	2014-07-30
1.2.1	2014-07-14
1.2.0	2014-07-11
1.1.1	2014-06-19
1.1.0	2014-06-12
1.0.1	2014-06-04

Examples

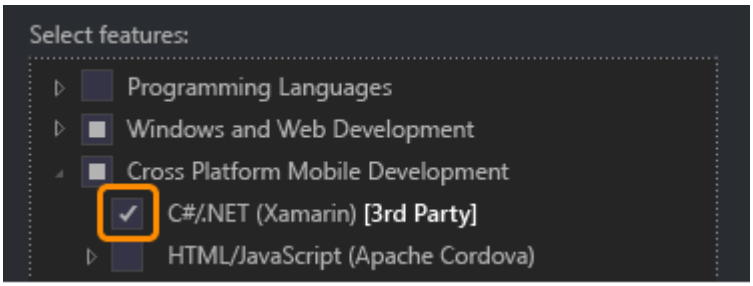
Installation (Visual Studio)

Xamarin.Forms ist eine plattformübergreifende, nativ gestützte UI-Toolkit-Abstraktion, mit der Entwickler auf einfache Weise Benutzeroberflächen erstellen können, die von Android, iOS, Windows und Windows Phone gemeinsam genutzt werden können. Die Benutzeroberflächen werden mithilfe der systemeigenen Steuerelemente der Zielplattform gerendert, sodass die Xamarin.Forms-Anwendungen für jede Plattform das richtige Erscheinungsbild beibehalten können.

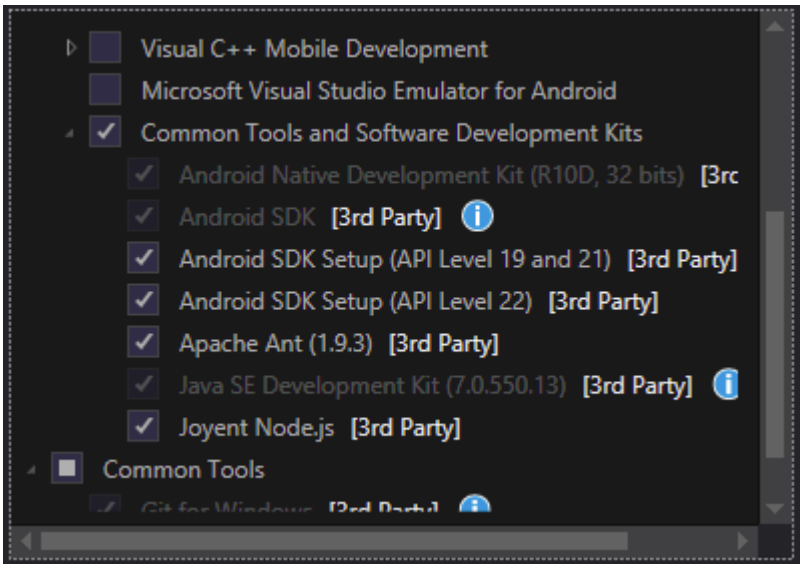
Xamarin Plugin für Visual Studio

Um mit Xamarin.Forms für Visual Studio zu beginnen, benötigen Sie das Xamarin-Plugin. Am einfachsten lässt es sich installieren, indem Sie das neueste Visual Studio herunterladen und installieren.

Wenn Sie bereits das neueste Visual Studio installiert haben, gehen Sie zu Systemsteuerung> Programme und Funktionen, klicken Sie mit der rechten Maustaste auf Visual Studio, und klicken Sie auf Ändern. Klicken Sie nach dem Öffnen des Installationsprogramms auf Ändern, und wählen Sie die plattformübergreifenden Tools für die mobile Entwicklung aus:



Sie können das Android SDK auch installieren:



Deaktivieren Sie es, wenn Sie das SDK bereits installiert haben. Sie können Xamarin später so einrichten, dass das vorhandene Android SDK verwendet wird.

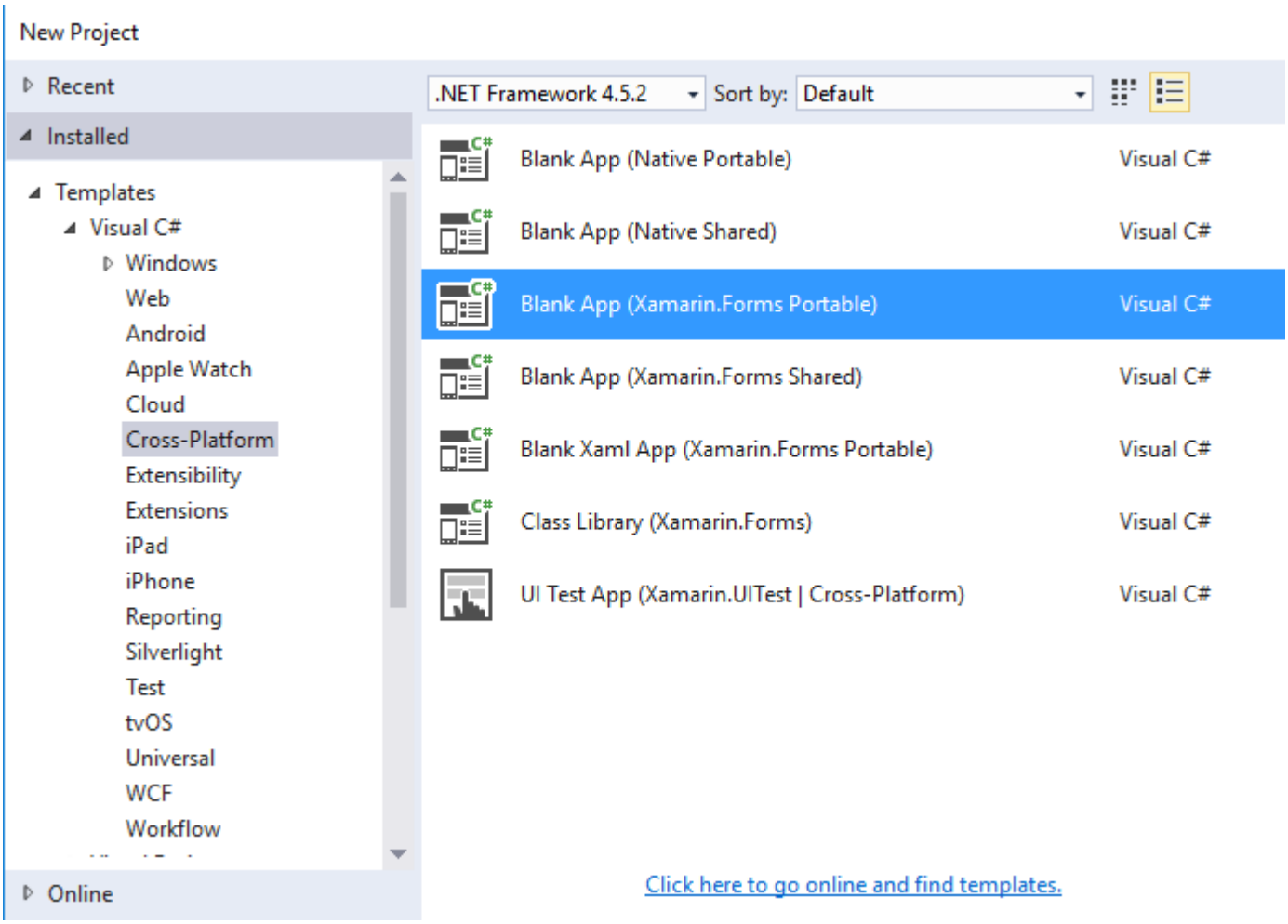
Xamarin.Forms

Xamarin.Forms ist ein Satz von Bibliotheken für Ihre Portable Class-Bibliothek und native Assemblys. Die Xamarin.Forms-Bibliothek selbst ist als NuGet-Paket verfügbar. Um es zu Ihrem Projekt hinzuzufügen, verwenden Sie einfach den regulären Befehl " `Install-Package` der Package Manager Console:

```
Install-Package Xamarin.Forms
```

für alle Ihre ursprünglichen Assemblys (z. B. MyProject, MyProject.Droid und MyProject.iOS).

Der einfachste Einstieg in Xamarin.Forms ist das Erstellen eines leeren Projekts in Visual Studio:



Wie Sie sehen, gibt es zwei Optionen zum Erstellen der leeren App - Portable und Shared. Ich empfehle Ihnen, mit Portable one zu beginnen, da es in der Realität am häufigsten verwendet wird (Unterschiede und weitere Erläuterungen sind hinzuzufügen).

Stellen Sie nach dem Erstellen des Projekts sicher, dass Sie die neueste Version von Xamarin.Forms verwenden, da Ihre ursprüngliche Vorlage möglicherweise die alte Version enthält. Verwenden Sie Ihre Package Manager Console oder die Option NuGet Packages verwalten, um ein Upgrade auf die neuesten Xamarin.Forms durchzuführen (denken Sie daran, es handelt sich lediglich um ein NuGet-Paket).

Während die Visual Studio Xamarin.Forms-Vorlagen ein iOS-Plattformprojekt für Sie erstellen, müssen Sie Xamarin mit einem Mac-Build-Host verbinden, um diese Projekte auf dem iOS-Simulator oder auf physischen Geräten ausführen zu können.

Hello World Xamarin Forms: Visual Studio

Nach der erfolgreichen Installation von Xamarin, wie im ersten Beispiel beschrieben, ist es Zeit, die erste Beispielanwendung zu starten.

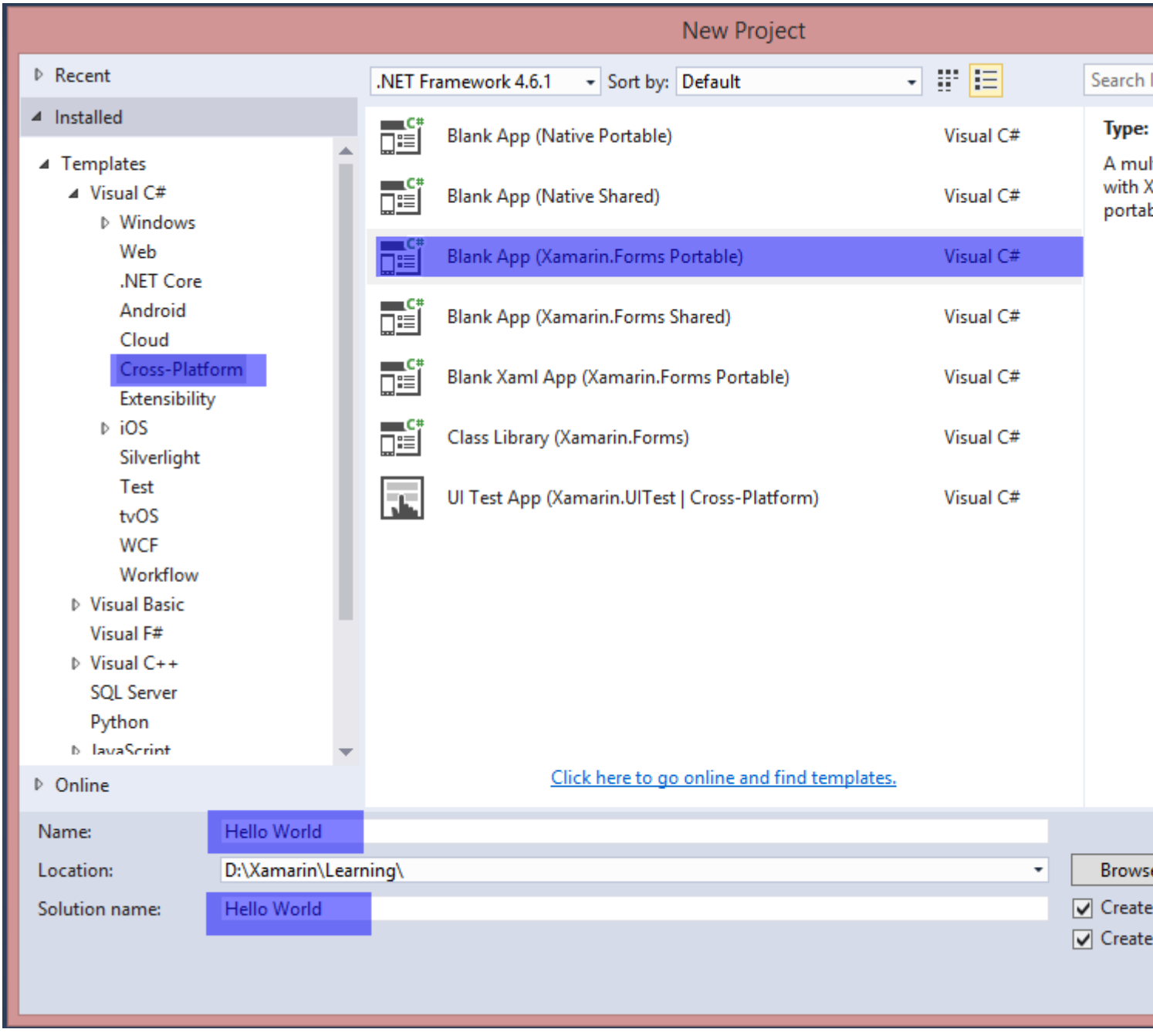
Schritt 1: Ein neues Projekt erstellen.

Wählen Sie in Visual Studio Neu -> Projekt -> Visual C # -> Plattformübergreifend -> Leere

Anwendung (Xamarin.Forms Portable).

Benennen Sie die App "Hello World" und wählen Sie den Ort, an dem das Projekt erstellt werden soll, und klicken Sie auf OK. Dadurch wird eine Lösung für Sie erstellt, die drei Projekte enthält:

1. HelloWorld (hier werden Ihre Logik und Ansichten platziert, dh das tragbare Projekt)
2. HelloWorld.Droid (das Android-Projekt)
3. HelloWorld.iOS (das iOS-Projekt)



Schritt 2: Untersuchung der Probe

Nachdem Sie die Lösung erstellt haben, kann eine Beispielanwendung bereitgestellt werden. Öffnen Sie die `App.cs` sich im Stammverzeichnis des tragbaren Projekts befindet, und untersuchen Sie den Code. Wie unten zu sehen, ist der `Content` des Samples ein `StackLayout` das ein `Label`

enthält:

```
using Xamarin.Forms;

namespace Hello_World
{
    public class App : Application
    {
        public App()
        {
            // The root page of your application
            MainPage = new ContentPage
            {
                Content = new StackLayout
                {
                    VerticalOptions = LayoutOptions.Center,
                    Children = {
                        new Label {
                            HorizontalTextAlignment = TextAlignment.Center,
                            Text = "Welcome to Xamarin Forms!"
                        }
                    }
                }
            };
        }
        protected override void OnStart()
        {
            // Handle when your app starts
        }
        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }
        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```

Schritt 3: Starten der Anwendung

Jetzt einfach mit der rechten Maustaste auf das Projekt , das Sie (starten wollen `HelloWorld.Droid` oder `HelloWorld.iOS`) und klicken Sie auf `Set as StartUp Project` . Klicken Sie dann in der Visual Studio-Symbolleiste auf die Schaltfläche `Start` (die grüne dreieckige Schaltfläche, die einer Wiedergabeschaltfläche ähnelt), um die Anwendung auf dem Zielsimulator / Emulator zu starten.

Erste Schritte mit Xamarin.Forms online lesen: https://riptutorial.com/de/xamarin-forms/topic/908/erste-schritte-mit-xamarin-forms

Kapitel 2: Abhängigkeitsdienste

Bemerkungen

Zugriff auf plattformspezifische API von PCL oder Shared Project

Examples

Greifen Sie auf Kamera und Galerie zu.

(Zugangscode) [https://github.com/vDoers/vDoersCameraAccess]

Abhängigkeitsdienste online lesen: https://riptutorial.com/de/xamarin-forms/topic/6127/abhangigkeitsdienste

Kapitel 3: Alert anzeigen

Examples

DisplayAlert

Ein Benachrichtigungsfeld kann auf einer `Xamarin.Forms Page` mit der Methode `DisplayAlert` . Wir können einen Titel, einen Text (zu alarmierender Text) und ein / zwei Aktionsschaltflächen bereitstellen. `Page` bietet zwei Überschreibungen der `DisplayAlert` Methode.

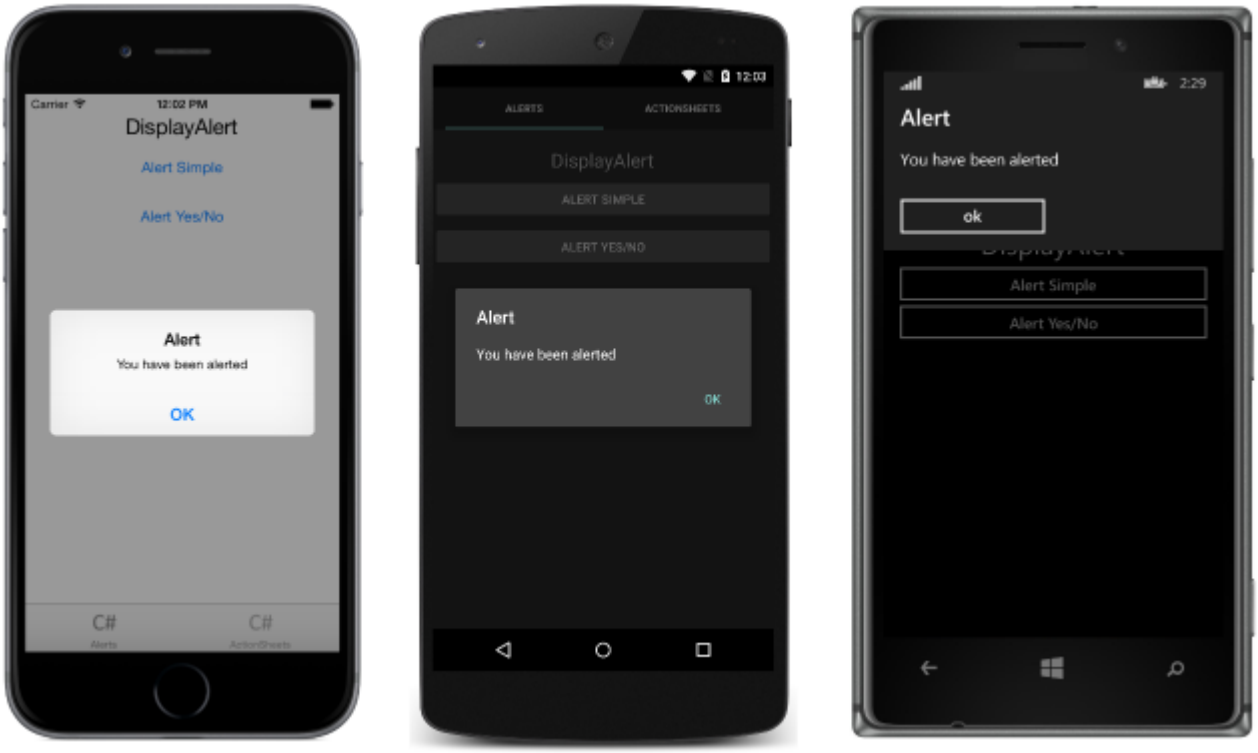
1. `public Task DisplayAlert (String title, String message, String cancel)`

Bei dieser Außerkraftsetzung wird dem Anwendungsbenutzer ein Warnungsdialogfeld mit einer einzigen Schaltfläche zum Abbrechen angezeigt. Die Warnung wird modal angezeigt. Sobald der Benutzer den Vorgang abgewiesen hat, interagiert er weiterhin mit der Anwendung.

Beispiel:

```
DisplayAlert ("Alert", "You have been alerted", "OK");
```

Das `AlertDialog` Snippet zeigt eine native Implementierung von Alerts auf jeder Plattform (`AlertDialog` in Android, `UIAlertView` in iOS, `MessageDialog` in Windows) wie `MessageDialog` .



2. `public System.Threading.Tasks.Task<bool> DisplayAlert (String title, String message, String accept, String cancel)`

Diese Überschreibung zeigt dem Anwendungsbenutzer einen Warnungsdialog mit einer Schaltfläche zum Annehmen und Abbrechen. Es erfasst die Antwort eines Benutzers, indem zwei

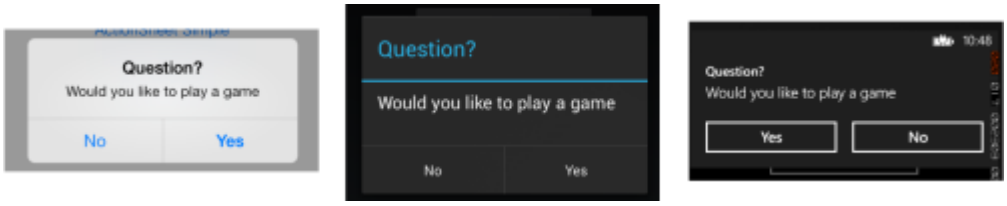
Schaltflächen angezeigt werden und ein `boolean` . Um eine Antwort von einer Warnung zu erhalten, geben Sie Text für beide Schaltflächen an und warten Sie auf die Methode. Nachdem der Benutzer eine der Optionen ausgewählt hat, wird die Antwort an den Code zurückgegeben.

Beispiel:

```
var answer = await DisplayAlert ("Question?", "Would you like to play a game", "Yes", "No");
Debug.WriteLine ("Answer: " + (answer?"Yes":"No"));
```

Beispiel 2: (Wenn Bedingung wahr oder falsch geprüft wird, um fortzufahren)

```
async void listSelected(object sender, SelectedItemChangedEventArgs e)
    {
        var ans = await DisplayAlert("Question?", "Would you like Delete", "Yes", "No");
        if (ans == true)
        {
            //Success condition
        }
        else
        {
            //false conditon
        }
     }
```



Alert-Beispiel mit nur einer Taste und Aktion

```
var alertResult = await DisplayAlert("Alert Title", Alert Message, null, "OK");
if(!alertResult)
{
   //do your stuff.
}
```

Hier bekommen wir eine OK-Klick-Aktion.

Alert anzeigen online lesen: https://riptutorial.com/de/xamarin-forms/topic/4883/alert-anzeigen

Kapitel 4: AppSettings Reader in Xamarin.Forms

Examples

Einlesen der app.config-Datei in einem Xamarin.Forms-Xaml-Projekt

Jede mobile Plattform bietet zwar eine eigene Einstellungsverwaltungs-API, aber es gibt keine eingebauten Möglichkeiten, Einstellungen aus einer guten alten XML-Datei mit dem Namen .net style app.config zu lesen. Dies ist auf eine Reihe von guten Gründen zurückzuführen, insbesondere darauf, dass das Konfigurations-API für .net Framework-Konfiguration schwergewichtig ist und jede Plattform über ein eigenes Dateisystem-API verfügt.

Deshalb haben wir eine einfache PCLAppConfig- Bibliothek erstellt, die für Ihren unmittelbaren Verbrauch gut verpackt ist.

Diese Bibliothek verwendet die schöne PCLStorage- Bibliothek

In diesem Beispiel wird davon ausgegangen, dass Sie ein Xamarin.Forms Xaml-Projekt entwickeln, in dem Sie auf Einstellungen in Ihrem freigegebenen Ansichtsmodell zugreifen müssen.

1. Initialisieren Sie ConfigurationManager.AppSettings für jedes Ihrer Plattformprojekt direkt nach der Anweisung 'Xamarin.Forms.Forms.Init' wie folgt:

iOS (AppDelegate.cs)

```
global::Xamarin.Forms.Forms.Init();
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

Android (Hauptaktivität.cs)

```
global::Xamarin.Forms.Forms.Init(this, bundle);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
LoadApplication(new App());
```

UWP / Windows 8.1 / WP 8.1 (App.xaml.cs)

```
Xamarin.Forms.Forms.Init(e);
ConfigurationManager.Initialise(PCLAppConfig.FileSystemStream.PortableStream.Current);
```

2. Fügen Sie Ihrem freigegebenen PCL-Projekt eine app.config-Datei hinzu, und fügen Sie Ihre appSettings-Einträge hinzu, wie Sie es auch mit jeder app.config-Datei tun würden

```
<configuration>
```

```
    <appSettings>
        <add key="config.text" value="hello from app.settings!" />
    </appSettings>
</configuration>
```

3. Fügen Sie diese PCL-Datei app.config in allen Ihren Plattformprojekten **als verknüpfte Datei hinzu** . **Stellen Sie** für Android sicher, dass die Build-Aktion auf **"AndroidAsset" gesetzt** ist. Für UWP setzen Sie die Build-Aktion auf **"Content".**

4. `ConfigurationManager.AppSettings["config.text"];` Ihre Einstellung zu:
   `ConfigurationManager.AppSettings["config.text"];`

AppSettings Reader in Xamarin.Forms online lesen: https://riptutorial.com/de/xamarin-forms/topic/5911/appsettings-reader-in-xamarin-forms

Kapitel 5: Auslöser und Verhalten

Examples

Xamarin Forms Trigger Beispiel

`Trigger` sind eine einfache Möglichkeit, Ihrer Anwendung etwas UX-Reaktionsfähigkeit zu verleihen. Ein einfacher Weg, dies zu tun, ist das Hinzufügen eines `Trigger` die `TextColor` eines `Label TextColor` , je nachdem, ob in den zugehörigen `Entry` Text eingegeben wurde.

Bei Verwendung eines `Trigger` kann `Label.TextColor` von grau (wenn kein Text eingegeben wird) in schwarz (sobald der Benutzer Text eingibt) ändern:

Wandler (jeder Wandler ein gegebene `Instance` - Variable , die in der Bindung verwendet wird , so dass eine neue Instanz der Klasse , die nicht jedes Mal erstellt wird es verwendet wird):

```
/// <summary>
/// Used in a XAML trigger to return <c>true</c> or <c>false</c> based on the length of
<c>value</c>.
/// </summary>
public class LengthTriggerConverter : Xamarin.Forms.IValueConverter {

    /// <summary>
    /// Used so that a new instance is not created every time this converter is used in the
XAML code.
    /// </summary>
    public static LengthTriggerConverter Instance = new LengthTriggerConverter();

    /// <summary>
    /// If a `ConverterParameter` is passed in, a check to see if <c>value</c> is greater than
<c>parameter</c> is made. Otherwise, a check to see if <c>value</c> is over 0 is made.
    /// </summary>
    /// <param name="value">The length of the text from an Entry/Label/etc.</param>
    /// <param name="targetType">The Type of object/control that the text/value is coming
from.</param>
    /// <param name="parameter">Optional, specify what length to test against (example: for 3
Letter Name, we would choose 2, since the 3 Letter Name Entry needs to be over 2 characters),
if not specified, defaults to 0.</param>
    /// <param name="culture">The current culture set in the device.</param>
    /// <returns><c>object</c>, which is a <c>bool</c> (<c>true</c> if <c>value</c> is greater
than 0 (or is greater than the parameter), <c>false</c> if not).</returns>
    public object Convert(object value, System.Type targetType, object parameter, CultureInfo
culture)     { return DoWork(value, parameter); }
    public object ConvertBack(object value, System.Type targetType, object parameter,
CultureInfo culture) { return DoWork(value, parameter); }

    private static object DoWork(object value, object parameter) {
        int parameterInt = 0;

        if(parameter != null) { //If param was specified, convert and use it, otherwise, 0 is
used

            string parameterString = (string)parameter;
```

```
            if(!string.IsNullOrEmpty(parameterString)) { int.TryParse(parameterString, out
parameterInt); }
        }

        return (int)value > parameterInt;
    }
}
```

XAML (der XAML - Code verwendet die `x:Name` des `Entry` im herauszufinden `Entry.Text` Eigenschaft ist mehr als 3 Zeichen lang sein .):

```
<StackLayout>
    <Label Text="3 Letter Name">
        <Label.Triggers>
            <DataTrigger TargetType="Label"
                         Binding="{Binding Source={x:Reference NameEntry},
                                           Path=Text.Length,
                                           Converter={x:Static
helpers:LengthTriggerConverter.Instance},
                                           ConverterParameter=2}"
                         Value="False">
                <Setter Property="TextColor"
                        Value="Gray"/>
            </DataTrigger>
        </Label.Triggers>
    </Label>
    <Entry x:Name="NameEntry"
           Text="{Binding MealAmount}"
           HorizontalOptions="StartAndExpand"/>
</StackLayout>
```

Multi-Trigger

MultiTrigger wird nicht häufig benötigt, aber es gibt Situationen, in denen es sehr praktisch ist. MultiTrigger verhält sich ähnlich wie Trigger oder DataTrigger, hat jedoch mehrere Bedingungen. Alle Bedingungen müssen erfüllt sein, damit ein Setter feuern kann. Hier ist ein einfaches Beispiel:

```
<!-- Text field needs to be initialized in order for the trigger to work at start -->
<Entry x:Name="email" Placeholder="Email" Text="" />
<Entry x:Name="phone" Placeholder="Phone" Text="" />
<Button Text="Submit">
    <Button.Triggers>
        <MultiTrigger TargetType="Button">
            <MultiTrigger.Conditions>
                <BindingCondition Binding="{Binding Source={x:Reference email},
Path=Text.Length}" Value="0" />
                <BindingCondition Binding="{Binding Source={x:Reference phone},
Path=Text.Length}" Value="0" />
            </MultiTrigger.Conditions>
            <Setter Property="IsEnabled" Value="False" />
        </MultiTrigger>
    </Button.Triggers>
</Button>
```

Das Beispiel enthält zwei verschiedene Einträge, Telefon und E-Mail, und einer davon muss ausgefüllt werden. Der MultiTrigger deaktiviert die Schaltfläche zum Senden, wenn beide Felder

leer sind.

Auslöser und Verhalten online lesen: https://riptutorial.com/de/xamarin-forms/topic/6012/ausloser-und-verhalten

Kapitel 6: Ausnahmebehandlung

Examples

Eine Möglichkeit, über iOS Ausnahmen zu berichten

Gehen Sie zur Datei " `Main.cs` " im **iOS-Projekt** und ändern Sie den vorhandenen Code (`Main.cs` unten):

```
    static void Main(string[] args)
    {
        try
        {
            UIApplication.Main(args, null, "AppDelegate");
        }
        catch (Exception ex)
        {
            Debug.WriteLine("iOS Main Exception: {0}", ex);

            var watson = new LittleWatson();
            watson.SaveReport(ex);
        }
    }
```

`ILittleWatson` in Portable Code verwendete `ILittleWatson` Schnittstelle könnte folgendermaßen aussehen:

```
public interface ILittleWatson
{
    Task<bool> SendReport();

    void SaveReport(Exception ex);
}
```

Implementierung für **iOS-Projekt** :

```
[assembly: Xamarin.Forms.Dependency(typeof(LittleWatson))]
namespace SomeNamespace
{
    public class LittleWatson : ILittleWatson
    {
        private const string FileName = "Report.txt";

        private readonly static string DocumentsFolder;
        private readonly static string FilePath;

        private TaskCompletionSource<bool> _sendingTask;

        static LittleWatson()
        {
            DocumentsFolder =
Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
            FilePath = Path.Combine(DocumentsFolder, FileName);
```

```
        }

        public async Task<bool> SendReport()
        {
            _sendingTask = new TaskCompletionSource<bool>();

            try
            {
                var text = File.ReadAllText(FilePath);
                File.Delete(FilePath);
                if (MFMailComposeViewController.CanSendMail)
                {
                    var email = ""; // Put receiver email here.
                    var mailController = new MFMailComposeViewController();
                    mailController.SetToRecipients(new string[] { email });
                    mailController.SetSubject("iPhone error");
                    mailController.SetMessageBody(text, false);
                    mailController.Finished += (object s, MFComposeResultEventArgs args) =>
                    {
                        args.Controller.DismissViewController(true, null);
                        _sendingTask.TrySetResult(true);
                    };

                    ShowViewController(mailController);
                }
            }
            catch (FileNotFoundException)
            {
                // No errors found.
                _sendingTask.TrySetResult(false);
            }

            return await _sendingTask.Task;
        }

        public void SaveReport(Exception ex)
        {
            var exceptionInfo = $"{ex.Message} - {ex.StackTrace}";
            File.WriteAllText(FilePath, exceptionInfo);
        }

        private static void ShowViewController(UIViewController controller)
        {
            var topController = UIApplication.SharedApplication.KeyWindow.RootViewController;
            while (topController.PresentedViewController != null)
            {
                topController = topController.PresentedViewController;
            }

            topController.PresentViewController(controller, true, null);
        }
    }
}
```

Und dann, wo die App beginnt, setzen Sie:

```
        var watson = DependencyService.Get<ILittleWatson>();
        if (watson != null)
        {
            await watson.SendReport();
```

```
        }
```

Ausnahmebehandlung online lesen: https://riptutorial.com/de/xamarin-forms/topic/6428/ausnahmebehandlung

Kapitel 7: Auswirkungen

Einführung

Effekte vereinfachen plattformspezifische Anpassungen. Wenn die Eigenschaften eines Xamarin Forms-Steuerelements geändert werden müssen, können Effekte verwendet werden. Wenn die Methoden des Xamarin Forms Control überschrieben werden müssen, können benutzerdefinierte Renderer verwendet werden

Examples

Plattformspezifischen Effekt für ein Entry-Control hinzufügen

1. Erstellen Sie eine neue Xamarin Forms App mit PCL-Datei -> Neue Lösung -> Plattformübergreifende App -> Xamarin Forms -> Forms App. `EffectsDemo` das Projekt als `EffectsDemo`
2. `OnElementPropertyChanged` Sie im iOS-Projekt eine neue `Effect` Klasse hinzu, die von der `PlatformEffect` Klasse erbt und die Methoden `OnAttached` , `OnDetached` und `OnElementPropertyChanged` `OnDetached` `OnElementPropertyChanged` Sie die beiden Attribute `ResolutionGroupName` und `ExportEffect` . Diese sind erforderlich, um diesen Effekt aus dem PCL / Shared-Projekt zu verwenden.

- `OnAttached` ist die Methode, bei der die Anpassungslogik verwendet wird
- `OnDetached` ist die Methode, bei der das Bereinigen und `OnDetached` der Registrierung erfolgt
- `OnElementPropertyChanged` ist die Methode, die bei Eigenschaftsänderungen verschiedener Elemente ausgelöst wird. Um die richtige Eigenschaft zu identifizieren, überprüfen Sie die genaue Eigenschaftsänderung und fügen Sie Ihre Logik hinzu. In diesem Beispiel gibt `OnFocus` die `Blue` Farbe und `OutofFocus` die `Red` Farbe

```
using System;
using EffectsDemo.iOS;
using UIKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ResolutionGroupName("xhackers")]
[assembly: ExportEffect(typeof(FocusEffect), "FocusEffect")]
namespace EffectsDemo.iOS
{
public class FocusEffect : PlatformEffect
{
public FocusEffect()
{
}
UIColor backgroundColor;
protected override void OnAttached()
{
```

```
     try
     {
         Control.BackgroundColor = backgroundColor = UIColor.Red;
     }
     catch (Exception ex)
     {
         Console.WriteLine("Cannot set attacked property" + ex.Message);
     }
 }

 protected override void OnDetached()
 {
     throw new NotImplementedException();
 }

 protected override void
OnElementPropertyChanged(System.ComponentModel.PropertyChangedEventArgs args)
 {
     base.OnElementPropertyChanged(args);

     try
     {
         if (args.PropertyName == "IsFocused")
         {
             if (Control.BackgroundColor == backgroundColor)
             {
                 Control.BackgroundColor = UIColor.Blue;
             }
             else
             {
                 Control.BackgroundColor = backgroundColor;
             }
         }
     }
     catch (Exception ex)
     {
         Console.WriteLine("Cannot set property " + ex.Message);
     }
 }
```

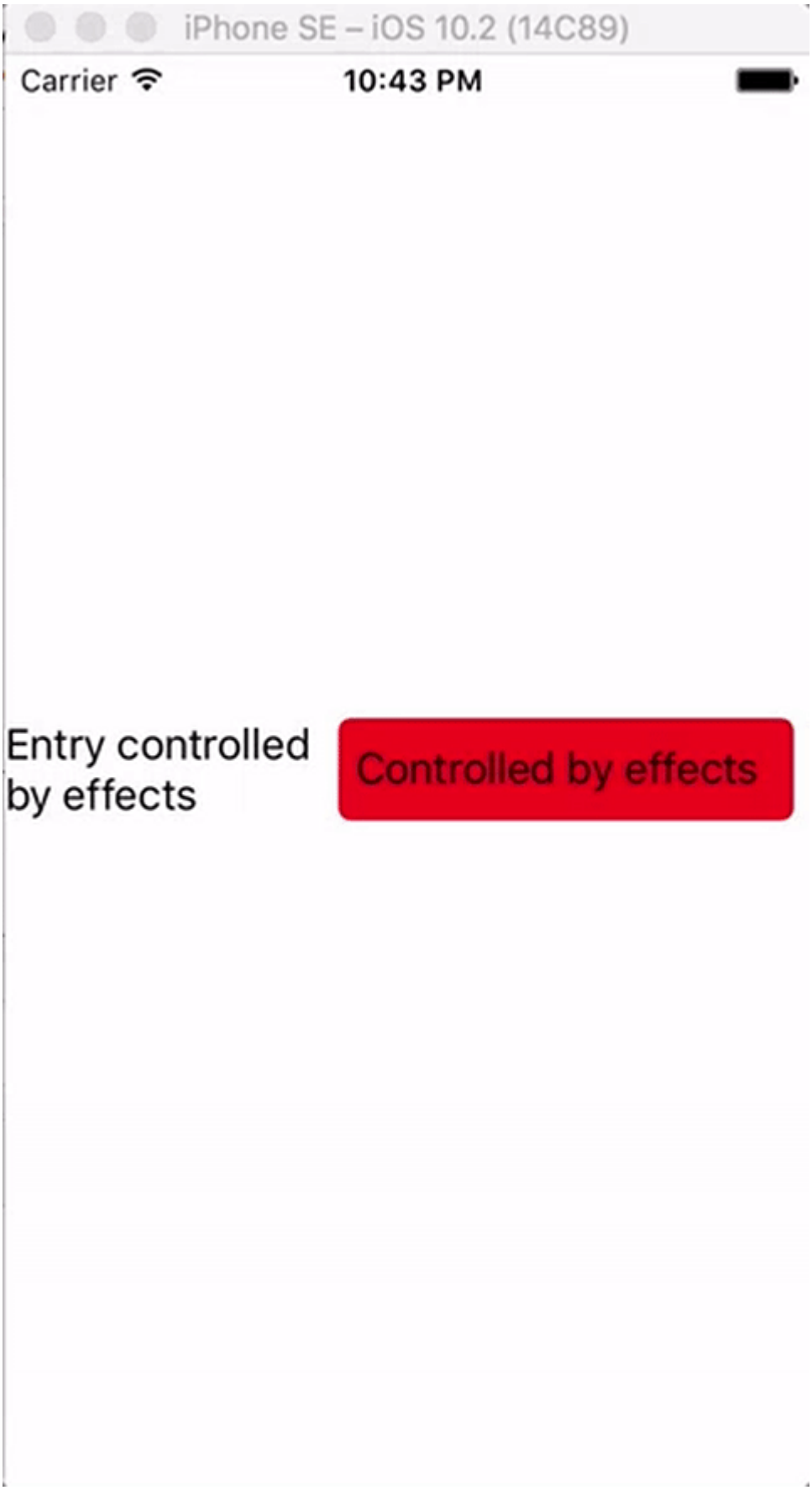
}}

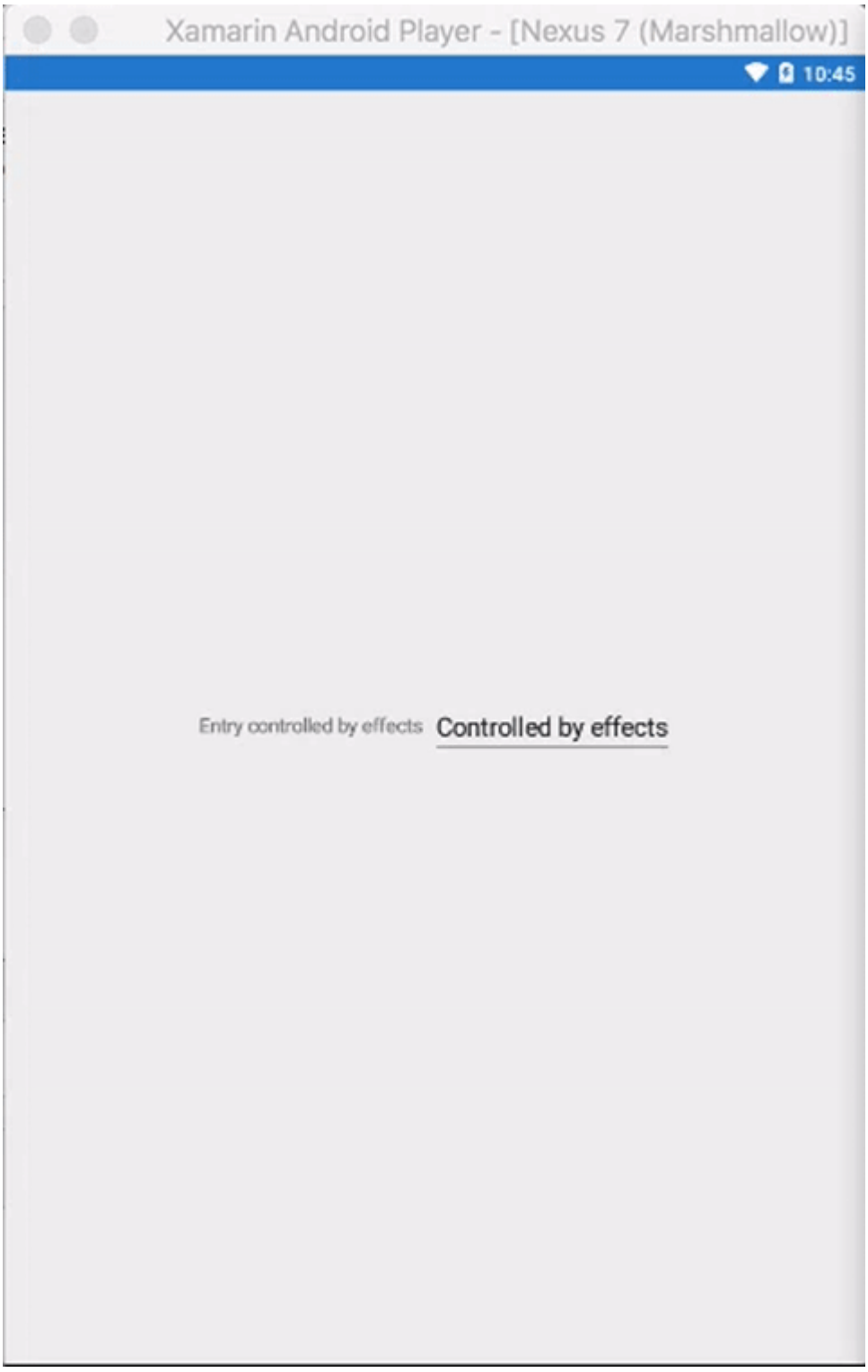
3. So verwenden Sie diesen Effekt in der Anwendung: Erstellen Sie im `PCL` Projekt eine neue Klasse mit dem Namen `FocusEffect` die von `RoutingEffect` erbt. Dies ist wichtig, damit die PCL die plattformspezifische Implementierung des Effekts instanziiert. Beispielcode unten:

```
using Xamarin.Forms;
namespace EffectsDemo
{
    public class FocusEffect : RoutingEffect
    {
        public FocusEffect() : base("xhackers.FocusEffect")
        {
        }
    }
}
```

4. Fügen Sie den Effekt zur `Entry` in der XAML hinzu

```
<?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:EffectsDemo" x:Class="EffectsDemo.EffectsDemoPage">
<StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
<Label Text="Effects Demo" HorizontalOptions="StartAndExpand" VerticalOptions="Center"
></Label>
<Entry Text="Controlled by effects" HorizontalOptions="FillAndExpand"
VerticalOptions="Center">
    <Entry.Effects>
        <local:FocusEffect>
        </local:FocusEffect>
    </Entry.Effects>
</Entry>
</StackLayout>
</ContentPage>
```


iPhone SE – iOS 10.2 (14C89)
Carrier
10:43 PM
Entry controlled by effects
Controlled by effects



Da war der Effekt nur in iOS - Version umgesetzt werden , wenn die Anwendung in läuft `iOS Simulator` die beim Fokussieren `Entry` Hintergrundfarbe ändert und nichts passiert in `Android Emulator` als der `Effect` nicht unter erstellt wurde `Droid` Projekt

Auswirkungen online lesen: https://riptutorial.com/de/xamarin-forms/topic/9252/auswirkungen

Kapitel 8: BDD-Unit-Tests in Xamarin.Forms

Bemerkungen

- Der DI-Container / Resolver, den wir intern in dieser Bibliothek verwenden, ist Autofac.
- Das Testframework ist NUnit 3x.
- Sie sollten diese Bibliothek mit jedem Xamarin.Forms-Framework verwenden können
- Quell- und Beispielprojekt hier verfügbar

Examples

Simple Specflow zum Testen von Befehlen und der Navigation mit NUnit Test Runner

Warum brauchen wir das?

Die aktuelle Methode für Unit-Tests in Xamarin.Forms ist über einen Plattformläufer. Daher muss Ihr Test in einer Ios-, Android-, Windows- oder Mac-Benutzeroberfläche ausgeführt werden: Ausführen von Tests in der IDE

Xamarin bietet auch großartige UI-Tests mit dem Xamarin.TestCloud- Angebot. Wenn BDD-Entwicklerpraktiken implementiert werden sollen und ViewModels und Befehle getestet werden können, während sie auf einem Unit-Testläufer lokal oder auf einem Build-Server kostengünstig ausgeführt werden, gibt es dies nicht in der Weise gebaut.

Ich habe eine Bibliothek entwickelt, die die Verwendung von Specflow mit Xamarin.Forms ermöglicht, um Ihre Features von Ihren Szenariodefinitionen bis zum ViewModel problemlos zu implementieren, unabhängig von allen für die App verwendeten MVVM-Frameworks (wie XLabs , MVVMCross , Prism).

Wenn Sie noch nicht mit BDD vertraut sind, aktivieren Sie " Specflow out".

Verwendungszweck:

- Wenn Sie es noch nicht haben, installieren Sie die specflow Visual Studio-Erweiterung von hier (oder von Ihrer Visual Studio-IDE): https://visualstudiogallery.msdn.microsoft.com/c74211e7-cb6e-4dfa-855d-df0ad4a37dd6
- Fügen Sie Ihrem Xamarin.Forms-Projekt eine Klassenbibliothek hinzu. Das ist dein Testprojekt.
- Fügen Sie das SpecFlow.Xamarin.Forms-Paket aus dem nuget zu Ihren Testprojekten hinzu.

- Fügen Sie Ihrem Testprojekt eine Klasse hinzu, die 'TestApp' erbt, und registrieren Sie Ihre Ansichten / Ansichtsmodellpaare sowie das Hinzufügen einer DI-Registrierung wie folgt:

```
public class DemoAppTest : TestApp
{
    protected override void SetViewModelMapping()
    {
        TestViewFactory.EnableCache = false;

        // register your views / viewmodels below
        RegisterView<MainPage, MainViewModel>();
    }

    protected override void InitialiseContainer()
    {
        // add any di registration here
        // Resolver.Instance.Register<TInterface, TType>();
        base.InitialiseContainer();
    }
}
```

- Fügen Sie Ihrem Testprojekt eine SetupHook-Klasse hinzu, um Ihre Specflow-Hooks hinzuzufügen. Sie müssen die Testanwendung wie folgt bootstrapieren, wobei Sie die oben erstellte Klasse und das anfängliche Viewmodel Ihrer App angeben:

```
[Binding]
public class SetupHooks : TestSetupHooks
{
    /// <summary>
    ///     The before scenario.
    /// </summary>
    [BeforeScenario]
    public void BeforeScenario()
    {
        // bootstrap test app with your test app and your starting viewmodel
        new TestAppBootstrap().RunApplication<DemoAppTest, MainViewModel>();
    }
}
```

- Sie müssen Ihrem xamarin.forms-Ansichten codebehind einen catch-Block hinzufügen, um das xamarin.forms-Framework zu ignorieren, das Sie dazu zwingt, die App ui auszuführen (etwas, das wir nicht möchten):

```
    public YourView()
    {
        try
        {
            InitializeComponent();
        }
        catch (InvalidOperationException soe)
        {
            if (!soe.Message.Contains("MUST"))
                throw;
        }
    }
```

- Fügen Sie Ihrem Projekt eine specflow-Funktion hinzu (mithilfe der vs specflow-Vorlagen, die im Lieferumfang der vs specflow-Erweiterung enthalten sind).
- Erstellen / Generieren Sie eine Schrittklasse, die TestStepBase erbt, und übergeben Sie den Parameter scenContext an die Basis.
- Verwenden Sie die Navigationsdienste und -helfer, um zu navigieren, Befehle auszuführen und Ihre Ansichtsmodelle zu testen:

```
  [Binding]
    public class GeneralSteps : TestStepBase
    {
        public GeneralSteps(ScenarioContext scenarioContext)
            : base(scenarioContext)
        {
            // you need to instantiate your steps by passing the scenarioContext to the base
        }

        [Given(@"I am on the main view")]
        public void GivenIAmOnTheMainView()
        {
            Resolver.Instance.Resolve<INavigationService>().PushAsync<MainViewModel>();


Resolver.Instance.Resolve<INavigationService>().CurrentViewModelType.ShouldEqualType<MainViewModel>();

        }

        [When(@"I click on the button")]
        public void WhenIClickOnTheButton()
        {
            GetCurrentViewModel<MainViewModel>().GetTextCommand.Execute(null);
        }

        [Then(@"I can see a Label with text ""(.*)""")]
        public void ThenICanSeeALabelWithText(string text)
        {
            GetCurrentViewModel<MainViewModel>().Text.ShouldEqual(text);
        }
    }
```

Erweiterte Verwendung für MVVM

Um das erste Beispiel hinzuzufügen, müssen wir dem ViewModel einen Haken für die Navigation geben, um Navigationsanweisungen zu testen, die in der Anwendung vorkommen. Um das zu erreichen:

- Fügen Sie das Paket SpecFlow.Xamarin.Forms.IViewModel aus Nuget Ihrem PCL Xamarin.Forms-Projekt hinzu
- Implementieren Sie die IViewModel-Schnittstelle in Ihrem ViewModel. Dadurch wird einfach die Xamarin.Forms INavigation-Eigenschaft angezeigt:
- `public class MainViewModel : INotifyPropertyChanged, IViewModel.IViewModel { public INavigation Navigation { get; set; }`
- Das Testframework wird das aufnehmen und die interne Navigation verwalten
- Sie können jedes MVVM-Framework für Ihre Anwendung verwenden (z. B. XLabs ,

MVVMCross , Prism), um nur einige zu nennen. Solange die IViewModel-Schnittstelle in Ihrem ViewModel implementiert ist, wird es vom Framework übernommen.

BDD-Unit-Tests in Xamarin.Forms online lesen: https://riptutorial.com/de/xamarin-forms/topic/6172/bdd-unit-tests-in-xamarin-forms

Kapitel 9: Benutzerdefinierte Renderer

Examples

Benutzerdefinierter Renderer für ListView

Mit benutzerdefinierten Renderern können Entwickler das Erscheinungsbild und das Verhalten der Xamarin.Forms-Steuerelemente auf jeder Plattform anpassen. Entwickler könnten Funktionen nativer Steuerelemente verwenden.

Zum Beispiel müssen wir das Blättern in `ListView` deaktivieren. Unter iOS ist `ListView` auch dann scrollbar, wenn alle Elemente auf dem Bildschirm angezeigt werden und der Benutzer die Liste nicht scrollen kann. `Xamarin.Forms.ListView` verwaltet diese Einstellung nicht. In diesem Fall hilft ein Renderer.

Erstens sollten wir ein benutzerdefiniertes Steuerelement in einem PCL-Projekt erstellen, das einige erforderliche bindbare Eigenschaften deklariert:

```
public class SuperListView : ListView
{
    public static readonly BindableProperty IsScrollingEnableProperty =
        BindableProperty.Create(nameof(IsScrollingEnable),
                                typeof(bool),
                                typeof(SuperListView),
                                true);

    public bool IsScrollingEnable
    {
        get { return (bool)GetValue(IsScrollingEnableProperty); }
        set { SetValue(IsScrollingEnableProperty, value); }
    }
}
```

Im nächsten Schritt wird für jede Plattform ein Renderer erstellt.

iOS:

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.iOS.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;


            Control.ScrollEnabled = superListView.IsScrollingEnable;
```

```
        }
    }
}
```

Und Android (Android-Liste hat keinen Bildlauf, wenn alle Elemente auf dem Bildschirm angezeigt werden. Daher wird das Bildlauf nicht deaktiviert. Trotzdem können wir native Eigenschaften verwenden):

```
[assembly: ExportRenderer(typeof(SuperListView), typeof(SuperListViewRenderer))]
namespace SuperForms.Droid.Renderers
{
    public class SuperListViewRenderer : ListViewRenderer
    {
        protected override void
OnElementChanged(ElementChangedEventArgs<Xamarin.Forms.ListView> e)
        {
            base.OnElementChanged(e);

            var superListView = Element as SuperListView;
            if (superListView == null)
                return;
        }
    }
}
```

`Element` des Renderers ist mein `SuperListView` Steuerelement aus dem PCL-Projekt.

`Control` Steuerelementeigenschaft des Renderers ist das native Steuerelement. `Android.Widget.ListView` für Android und `UIKit.UITableView` für iOS.

Und wie wir es in `XAML` :

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.SuperListViewPage">


  <controls:SuperListView ItemsSource="{Binding Items, Source={x:Reference Page}}"
                          IsScrollingEnable="false"
                          Margin="20">
    <controls:SuperListView.ItemTemplate>
      <DataTemplate>
        <ViewCell>
          <Label Text="{Binding .}"/>
        </ViewCell>
      </DataTemplate>
    </controls:SuperListView.ItemTemplate>
  </controls:SuperListView>
</ContentPage>
```

`.cs` Datei der Seite:

```
public partial class SuperListViewPage : ContentPage
{
```

```
    private ObservableCollection<string> _items;

    public ObservableCollection<string> Items
    {
        get { return _items; }
        set
        {
            _items = value;
            OnPropertyChanged();
        }
    }

    public SuperListViewPage()
    {
        var list = new SuperListView();

        InitializeComponent();

        var items = new List<string>(10);
        for (int i = 1; i <= 10; i++)
        {
            items.Add($"Item {i}");
        }

        Items = new ObservableCollection<string>(items);
    }
}
```

Benutzerdefinierter Renderer für BoxView

Mit Hilfe der benutzerdefinierten Renderer-Hilfe können Sie neue Eigenschaften hinzufügen und auf der nativen Plattform anders darstellen, als dies durch gemeinsam genutzten Code möglich ist. In diesem Beispiel werden Radien und Schatten zu einer Boxansicht hinzugefügt.

Erstens sollten wir ein benutzerdefiniertes Steuerelement in einem PCL-Projekt erstellen, das einige erforderliche bindbare Eigenschaften deklariert:

```
namespace Mobile.Controls
{
    public class ExtendedBoxView : BoxView
    {
        /// <summary>
        /// Respresents the background color of the button.
        /// </summary>
        public static readonly BindableProperty BorderRadiusProperty =
BindableProperty.Create<ExtendedBoxView, double>(p => p.BorderRadius, 0);

        public double BorderRadius
        {
            get { return (double)GetValue(BorderRadiusProperty); }
            set { SetValue(BorderRadiusProperty, value); }
        }

        public static readonly BindableProperty StrokeProperty =
            BindableProperty.Create<ExtendedBoxView, Color>(p => p.Stroke, Color.Transparent);

        public Color Stroke
        {
```

```
            get { return (Color)GetValue(StrokeProperty); }
            set { SetValue(StrokeProperty, value); }
        }

        public static readonly BindableProperty StrokeThicknessProperty =
            BindableProperty.Create<ExtendedBoxView, double>(p => p.StrokeThickness, 0);

        public double StrokeThickness
        {
            get { return (double)GetValue(StrokeThicknessProperty); }
            set { SetValue(StrokeThicknessProperty, value); }
        }
    }
}
```

Im nächsten Schritt wird für jede Plattform ein Renderer erstellt.

iOS:

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.iOS.Renderers
{
 public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
 {
    public ExtendedBoxViewRenderer()
    {
    }

    protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
    {
        base.OnElementChanged(e);
        if (Element == null)
            return;

        Layer.MasksToBounds = true;
        Layer.CornerRadius = (float)((ExtendedBoxView)this.Element).BorderRadius / 2.0f;
    }


    protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        base.OnElementPropertyChanged(sender, e);
        if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
        {
            SetNeedsDisplay();
        }
    }

    public override void Draw(CGRect rect)
    {
        ExtendedBoxView roundedBoxView = (ExtendedBoxView)this.Element;
        using (var context = UIGraphics.GetCurrentContext())
        {
            context.SetFillColor(roundedBoxView.Color.ToCGColor());
            context.SetStrokeColor(roundedBoxView.Stroke.ToCGColor());
            context.SetLineWidth((float)roundedBoxView.StrokeThickness);

            var rCorner = this.Bounds.Inset((int)roundedBoxView.StrokeThickness / 2,
(int)roundedBoxView.StrokeThickness / 2);
```

```
            nfloat radius = (nfloat)roundedBoxView.BorderRadius;
            radius = (nfloat)Math.Max(0, Math.Min(radius, Math.Max(rCorner.Height / 2,
rCorner.Width / 2)));

            var path = CGPath.FromRoundedRect(rCorner, radius, radius);
            context.AddPath(path);
            context.DrawPath(CGPathDrawingMode.FillStroke);
        }
    }

 }
}
```

Sie können wieder anpassen, wie Sie möchten, innerhalb der Zeichenmethode.

Und das gleiche für Android:

```
[assembly: ExportRenderer(typeof(ExtendedBoxView), typeof(ExtendedBoxViewRenderer))]
namespace Mobile.Droid
{
    /// <summary>
    ///
    /// </summary>
    public class ExtendedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        /// <summary>
        ///
        /// </summary>
        public ExtendedBoxViewRenderer()
        {
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="e"></param>
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            SetWillNotDraw(false);

            Invalidate();
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedBoxView.BorderRadiusProperty.PropertyName)
            {
                Invalidate();
            }
```

```
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="canvas"></param>
        public override void Draw(Canvas canvas)
        {
            var box = Element as ExtendedBoxView;
            base.Draw(canvas);
            Paint myPaint = new Paint();

            myPaint.SetStyle(Paint.Style.Stroke);
            myPaint.StrokeWidth = (float)box.StrokeThickness;
            myPaint.SetARGB(convertTo255ScaleColor(box.Color.A),
convertTo255ScaleColor(box.Color.R), convertTo255ScaleColor(box.Color.G),
convertTo255ScaleColor(box.Color.B));
            myPaint.SetShadowLayer(20, 0, 5, Android.Graphics.Color.Argb(100, 0, 0, 0));

            SetLayerType(Android.Views.LayerType.Software, myPaint);

            var number = (float)box.StrokeThickness / 2;
            RectF rectF = new RectF(
                        number, // left
                        number, // top
                        canvas.Width - number, // right
                        canvas.Height - number // bottom
                );


            var radius = (float)box.BorderRadius;
            canvas.DrawRoundRect(rectF, radius, radius, myPaint);
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="color"></param>
        /// <returns></returns>
        private int convertTo255ScaleColor(double color)
        {
            return (int) Math.Ceiling(color * 255);
        }
}
```

}

Die XAML:

Wir beziehen uns zunächst auf unsere Kontrolle mit dem zuvor definierten Namespace.

```
xmlns:Controls="clr-namespace:Mobile.Controls"
```

Wir verwenden dann das Control wie folgt und verwenden Eigenschaften, die zu Beginn definiert wurden:

```
<Controls:ExtendedBoxView
    x:Name="search_boxview"
```

```
    Color="#444"
    BorderRadius="5"
    HorizontalOptions="CenterAndExpand"
    />
```

Auf den Renderer von einem nativen Projekt aus zugreifen

```
var renderer = Platform.GetRenderer(visualElement);

if (renderer == null)
{
    renderer = Platform.CreateRenderer(visualElement);
    Platform.SetRenderer(visualElement, renderer);
}

DoSomeThingWithRender(render); // now you can do whatever you want with render
```

Abgerundetes Label mit einem benutzerdefinierten Renderer für Frame (PCL- und iOS-Teile)

Erster Schritt: PCL-Teil

```
using Xamarin.Forms;

namespace ProjectNamespace
{
    public class ExtendedFrame : Frame
    {
        /// <summary>
        /// The corner radius property.
        /// </summary>
        public static readonly BindableProperty CornerRadiusProperty =
            BindableProperty.Create("CornerRadius", typeof(double), typeof(ExtendedFrame),
0.0);

        /// <summary>
        /// Gets or sets the corner radius.
        /// </summary>
        public double CornerRadius
        {
            get { return (double)GetValue(CornerRadiusProperty); }
            set { SetValue(CornerRadiusProperty, value); }
        }
    }
}
```

Zweiter Schritt: iOS-Teil

```
using ProjectNamespace;
using ProjectNamespace.iOS;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ExportRenderer(typeof(ExtendedFrame), typeof(ExtendedFrameRenderer))]
namespace ProjectNamespace.iOS
```

```
{
    public class ExtendedFrameRenderer : FrameRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<Frame> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.MasksToBounds = true;
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName == ExtendedFrame.CornerRadiusProperty.PropertyName)
            {
                Layer.CornerRadius = (float)(Element as ExtendedFrame).CornerRadius;
            }
        }
    }
}
```

Dritter Schritt: XAML-Code zum Aufrufen eines ExtendedFrame

Wenn Sie es in einem XAML-Teil verwenden möchten, vergessen Sie nicht, Folgendes zu schreiben:

```
xmlns:controls="clr-namespace:ProjectNamespace;assembly:ProjectNamespace"
```

nach dem

```
xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
```

Jetzt können Sie den ExtendedFrame folgendermaßen verwenden:

```
<controls:ExtendedFrame
    VerticalOptions="FillAndExpand"
    HorizontalOptions="FillAndExpand"
    BackgroundColor="Gray"
    CornerRadius="35.0">
    <Frame.Content>
        <Label
            Text="MyText"
            TextColor="Blue"/>
    </Frame.Content>
</controls:ExtendedFrame>
```

Abgerundete BoxView mit wählbarer Hintergrundfarbe

Erster Schritt: PCL-Teil

```
public class RoundedBoxView : BoxView
{
    public static readonly BindableProperty CornerRadiusProperty =
        BindableProperty.Create("CornerRadius", typeof(double), typeof(RoundedEntry),
default(double));

    public double CornerRadius
    {
        get
        {
            return (double)GetValue(CornerRadiusProperty);
        }
        set
        {
            SetValue(CornerRadiusProperty, value);
        }
    }

    public static readonly BindableProperty FillColorProperty =
        BindableProperty.Create("FillColor", typeof(string), typeof(RoundedEntry),
default(string));

    public string FillColor
    {
        get
        {
            return (string) GetValue(FillColorProperty);
        }
        set
        {
            SetValue(FillColorProperty, value);
        }
    }
}
```

Zweiter Schritt: Droid-Teil

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.Droid
{
    public class RoundedBoxViewRenderer : VisualElementRenderer<BoxView>
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);
            SetWillNotDraw(false);
            Invalidate();
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);
            SetWillNotDraw(false);
            Invalidate();
        }
```

```
        public override void Draw(Canvas canvas)
        {
            var box = Element as RoundedBoxView;
            var rect = new Rect();
            var paint = new Paint
            {
                Color = Xamarin.Forms.Color.FromHex(box.FillColor).ToAndroid(),
                AntiAlias = true,
            };

            GetDrawingRect(rect);

            var radius = (float)(rect.Width() / box.Width * box.CornerRadius);

            canvas.DrawRoundRect(new RectF(rect), radius, radius, paint);
        }
    }
}
```

Dritter Schritt: iOS-Teil

```
[assembly: ExportRenderer(typeof(RoundedBoxView), typeof(RoundedBoxViewRenderer))]
namespace MyNamespace.iOS
{
    public class RoundedBoxViewRenderer : BoxRenderer
    {
        protected override void OnElementChanged(ElementChangedEventArgs<BoxView> e)
        {
            base.OnElementChanged(e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = Color.FromHex((Element as
RoundedBoxView).FillColor).ToCGColor();
            }
        }

        protected override void OnElementPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (Element != null)
            {
                Layer.CornerRadius = (float)(Element as RoundedBoxView).CornerRadius;
                Layer.BackgroundColor = (Element as RoundedBoxView).FillColor.ToCGColor();
            }
        }
    }
}
```

Benutzerdefinierte Renderer online lesen: https://riptutorial.com/de/xamarin-forms/topic/2949/benutzerdefinierte-renderer

Kapitel 10: Benutzerdefinierte Schriftarten in Stilen

Bemerkungen

Ressourcen zum Anschauen:

- Xamarin Styles
- Verwenden von benutzerdefinierten Schriftarten auf iOS und Android mit Xamarin.Forms
- Benutzerdefinierte Renderer
- Ressourcenwörterbücher
- Angehängte Eigenschaften

Examples

Zugriff auf benutzerdefinierte Schriftarten in Syles

Xamarin.Forms bieten einen hervorragenden Mechanismus zum Gestalten Ihrer plattformübergreifenden Anwendung mit globalen Stilen.

In der mobilen Welt muss Ihre Anwendung hübsch sein und sich von den anderen Anwendungen abheben. Eines dieser Zeichen sind benutzerdefinierte Schriftarten, die in der Anwendung verwendet werden.

Mit Power-Unterstützung für XAML Styling in Xamarin.Forms hat soeben einen Basisstil für alle Etiketten mit Ihren benutzerdefinierten Schriftarten erstellt.

Um benutzerdefinierte Schriftarten in Ihr iOS- und Android-Projekt aufzunehmen, folgen Sie der Anleitung in Benutzerdefinierte Schriftarten unter iOS und Android mit Xamarin.Forms- Beitrag von Gerald verwenden.

Deklarieren Sie den Stil im Ressourcenabschnitt der App.xaml-Datei. Dadurch werden alle Stile global sichtbar.

Von Geralds Beitrag oben müssen wir die StyleId-Eigenschaft verwenden, aber diese Eigenschaft ist nicht bindbar. Um sie in Style Setter verwenden zu können, müssen wir dazu eine Attachable-Eigenschaft erstellen:

```
public static class FontHelper
{
    public static readonly BindableProperty StyleIdProperty =
        BindableProperty.CreateAttached(
            propertyName: nameof(Label.StyleId),
            returnType: typeof(String),
            declaringType: typeof(FontHelper),
            defaultValue: default(String),
```

```
            propertyChanged: OnItemTappedChanged);

    public static String GetStyleId(BindableObject bindable) =>
(String)bindable.GetValue(StyleIdProperty);

    public static void SetStyleId(BindableObject bindable, String value) =>
bindable.SetValue(StyleIdProperty, value);


    public static void OnItemTappedChanged(BindableObject bindable, object oldValue, object
newValue)
    {
        var control = bindable as Element;
        if (control != null)
        {
            control.StyleId = GetStyleId(control);
        }
    }
}
```

Dann fügen Sie den Stil in der App.xaml-Ressource hinzu:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:h="clr-namespace:My.Helpers"
             x:Class="My.App">

  <Application.Resources>

    <ResourceDictionary>
        <Style x:Key="LabelStyle" TargetType="Label">
            <Setter Property="FontFamily" Value="Metric Bold" />
            <Setter Property="h:FontHelper.StyleId" Value="Metric-Bold" />
        </Style>
    </ResourceDictionary>

  </Application.Resources>

</Application>
```

Gemäß dem obigen Beitrag müssen wir einen benutzerdefinierten Renderer für Label erstellen, der von LabelRenderer On Android-Plattform erbt.

```
internal class LabelExRenderer : LabelRenderer
{
    protected override void OnElementChanged(ElementChangedEventArgs<Label> e)
    {
        base.OnElementChanged(e);
        if (!String.IsNullOrEmpty(e.NewElement?.StyleId))
        {
            var font = Typeface.CreateFromAsset(Forms.Context.ApplicationContext.Assets,
e.NewElement.StyleId + ".ttf");
            Control.Typeface = font;
        }
    }
}
```

Für die iOS-Plattform sind keine benutzerdefinierten Renderer erforderlich.

Jetzt können Sie Stil in Ihrer Seitenmarkierung erhalten:

Für spezifische Beschriftung

```
<Label Text="Some text" Style={StaticResource LabelStyle} />
```

Oder wenden Sie den Stil auf alle Beschriftungen auf der Seite an, indem Sie einen auf LabesStyle basierenden Stil erstellen

```
<!-- language: xaml -->

<?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="My.MainPage">

  <ContentPage.Resources>

    <ResourceDictionary>
        <Style TargetType="Label" BasedOn={StaticResource LabelStyle}>
        </Style>
    </ResourceDictionary>

  </ContentPage.Resources>

  <Label Text="Some text" />

</ContentPage>
```

Benutzerdefinierte Schriftarten in Stilen online lesen: https://riptutorial.com/de/xamarin-forms/topic/4854/benutzerdefinierte-schriftarten-in-stilen

Kapitel 11: Benutzerdefinierte Steuerelemente erstellen

Examples

Erstellen Sie ein benutzerdefiniertes Steuerelement für Xamarin Forms (keine native Eingabe erforderlich).

Nachfolgend finden Sie ein Beispiel für ein reines benutzerdefiniertes Steuerelement von Xamarin Forms. Hierfür wird kein benutzerdefiniertes Rendering ausgeführt, könnte jedoch leicht in meinem eigenen Code implementiert werden. Ich verwende dieses Steuerelement zusammen mit einem benutzerdefinierten Renderer für `Label` und `Entry` .

Die individuelle Steuerung ist ein `ContentView` mit einem `Label` , `Entry` , und ein `BoxView` innerhalb es an Ort und Stelle gehalten wird , unter Verwendung von 2 `StackLayout` s. Wir definieren auch mehrere bindbare Eigenschaften sowie ein `TextChanged` Ereignis.

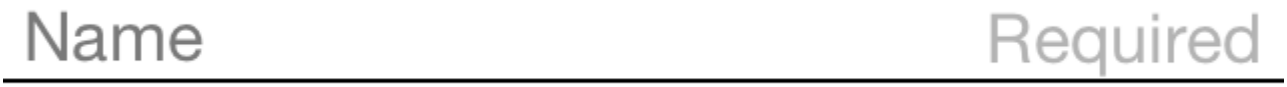
Die benutzerdefinierten bindbaren Eigenschaften funktionieren, indem sie wie folgt definiert werden und die Elemente innerhalb des Steuerelements (in diesem Fall eine `Label` und ein `Entry`) an die benutzerdefinierten bindbaren Eigenschaften gebunden sind. Einige Eigenschaften der bindbaren Eigenschaften müssen auch ein `BindingPropertyChangedDelegate` implementieren, damit die gebundenen Elemente ihre Werte ändern.

```
public class InputFieldContentView : ContentView {

    #region Properties

    /// <summary>
    /// Attached to the <c>InputFieldContentView</c>'s <c>ExtendedEntryOnTextChanged()</c>
event, but returns the <c>sender</c> as <c>InputFieldContentView</c>.
    /// </summary>
    public event System.EventHandler<TextChangedEventArgs> OnContentViewTextChangedEvent; //In
OnContentViewTextChangedEvent() we return our custom InputFieldContentView control as the
sender but we could have returned the Entry itself as the sender if we wanted to do that
instead.

    public static readonly BindableProperty LabelTextProperty =
BindableProperty.Create("LabelText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string LabelText {
        get { return (string)GetValue(LabelTextProperty); }
        set { SetValue(LabelTextProperty, value); }
    }

    public static readonly BindableProperty LabelColorProperty =
BindableProperty.Create("LabelColor", typeof(Color), typeof(InputFieldContentView),
Color.Default);

    public Color LabelColor {
```

```
        get { return (Color)GetValue(LabelColorProperty); }
        set { SetValue(LabelColorProperty, value); }
    }

    public static readonly BindableProperty EntryTextProperty =
BindableProperty.Create("EntryText", typeof(string), typeof(InputFieldContentView),
string.Empty, BindingMode.TwoWay, null, OnEntryTextChanged);

    public string EntryText {
        get { return (string)GetValue(EntryTextProperty); }
        set { SetValue(EntryTextProperty, value); }
    }

    public static readonly BindableProperty PlaceholderTextProperty =
BindableProperty.Create("PlaceholderText", typeof(string), typeof(InputFieldContentView),
string.Empty);

    public string PlaceholderText {
        get { return (string)GetValue(PlaceholderTextProperty); }
        set { SetValue(PlaceholderTextProperty, value); }
    }

    public static readonly BindableProperty UnderlineColorProperty =
BindableProperty.Create("UnderlineColor", typeof(Color), typeof(InputFieldContentView),
Color.Black, BindingMode.TwoWay, null, UnderlineColorChanged);

    public Color UnderlineColor {
        get { return (Color)GetValue(UnderlineColorProperty); }
        set { SetValue(UnderlineColorProperty, value); }
    }

    private BoxView _underline;

    #endregion

    public InputFieldContentView() {

        BackgroundColor   = Color.Transparent;
        HorizontalOptions = LayoutOptions.FillAndExpand;

        Label label = new Label {
            BindingContext    = this,
            HorizontalOptions = LayoutOptions.StartAndExpand,
            VerticalOptions   = LayoutOptions.Center,
            TextColor         = Color.Black
        };

        label.SetBinding(Label.TextProperty, (InputFieldContentView view) => view.LabelText,
BindingMode.TwoWay);
        label.SetBinding(Label.TextColorProperty, (InputFieldContentView view) =>
view.LabelColor, BindingMode.TwoWay);

        Entry entry = new Entry {
            BindingContext          = this,
            HorizontalOptions       = LayoutOptions.End,
            TextColor               = Color.Black,
            HorizontalTextAlignment = TextAlignment.End
        };

        entry.SetBinding(Entry.PlaceholderProperty, (InputFieldContentView view) =>
view.PlaceholderText, BindingMode.TwoWay);
```

```
        entry.SetBinding(Entry.TextProperty, (InputFieldContentView view) => view.EntryText,
BindingMode.TwoWay);

        entry.TextChanged += OnTextChangedEvent;

        _underline = new BoxView {
            BackgroundColor   = Color.Black,
            HeightRequest     = 1,
            HorizontalOptions = LayoutOptions.FillAndExpand
        };

        Content = new StackLayout {
            Spacing           = 0,
            HorizontalOptions = LayoutOptions.FillAndExpand,
            Children          = {
                new StackLayout {
                    Padding           = new Thickness(5, 0),
                    Spacing           = 0,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    Orientation       = StackOrientation.Horizontal,
                    Children          = { label, entry }
                }, _underline
            }
        };

        SizeChanged += (sender, args) => entry.WidthRequest = Width * 0.5 - 10;
    }

    private static void OnEntryTextChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView = (InputFieldContentView)bindable;
        contentView.EntryText             = (string)newValue;
    }

    private void OnTextChangedEvent(object sender, TextChangedEventArgs args) {
        if(OnContentViewTextChangedEvent != null) { OnContentViewTextChangedEvent(this, new
TextChangedEventArgs(args.OldTextValue, args.NewTextValue)); } //Here is where we pass in
'this' (which is the InputFieldContentView) instead of 'sender' (which is the Entry control)
    }

    private static void UnderlineColorChanged(BindableObject bindable, object oldValue, object
newValue) {
        InputFieldContentView contentView      = (InputFieldContentView)bindable;
        contentView._underline.BackgroundColor = (Color)newValue;
    }
}
```

Und hier ist ein Bild des Endprodukts unter iOS (das Bild zeigt, wie es aussieht, wenn ein benutzerdefinierter Renderer für `Label` und `Entry` wird, der zum Entfernen des Rahmens unter iOS und zum Festlegen einer benutzerdefinierten Schriftart für beide Elemente verwendet wird):



Ein Problem, dem ich `BoxView.BackgroundColor` , war, `BoxView.BackgroundColor` zu ändern, wenn sich `UnderlineColor` änderte. Selbst nach dem Binden der `BackgroundColor` Eigenschaft von `BoxView` wird diese `BoxView` nicht geändert, bis ich den `UnderlineColorChanged` Delegaten hinzugefügt habe.

Beschriftung mit bindbarer Sammlung von Feldern

Ich habe ein benutzerdefiniertes Label mit Wrapper für die `FormattedText` Eigenschaft erstellt:

```
public class MultiComponentLabel : Label
{
    public IList<TextComponent> Components { get; set; }

    public MultiComponentLabel()
    {
        var components = new ObservableCollection<TextComponent>();
        components.CollectionChanged += OnComponentsChanged;
        Components = components;
    }

    private void OnComponentsChanged(object sender, NotifyCollectionChangedEventArgs e)
    {
        BuildText();
    }

    private void OnComponentPropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        BuildText();
    }

    private void BuildText()
    {
        var formattedString = new FormattedString();
        foreach (var component in Components)
        {
            formattedString.Spans.Add(new Span { Text = component.Text });
            component.PropertyChanged -= OnComponentPropertyChanged;
            component.PropertyChanged += OnComponentPropertyChanged;
        }

        FormattedText = formattedString;
    }
}
```

Ich habe eine Sammlung von benutzerdefinierten `TextComponent` hinzugefügt:

```
public class TextComponent : BindableObject
{
    public static readonly BindableProperty TextProperty =
        BindableProperty.Create(nameof(Text),
                                typeof(string),
                                typeof(TextComponent),
                                default(string));

    public string Text
    {
        get { return (string)GetValue(TextProperty); }
        set { SetValue(TextProperty, value); }
    }
}
```

Und wenn Sammlung von Textkomponenten Änderungen oder `Text` - Eigenschaft separater

Komponente Änderungen , die ich wieder aufbauen `FormattedText` Eigentum von Basis `Label` .

Und wie ich es in `XAML` :

```
<ContentPage x:Name="Page"
         xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:controls="clr-namespace:SuperForms.Controls;assembly=SuperForms.Controls"
         x:Class="SuperForms.Samples.MultiComponentLabelPage">
<controls:MultiComponentLabel Margin="0,20,0,0">
    <controls:MultiComponentLabel.Components>
      <controls:TextComponent Text="Time"/>
      <controls:TextComponent Text=": "/>
      <controls:TextComponent Text="{Binding CurrentTime, Source={x:Reference Page}}"/>
    </controls:MultiComponentLabel.Components>
  </controls:MultiComponentLabel>
</ContentPage>
```

Codebehind der Seite:

```
public partial class MultiComponentLabelPage : ContentPage
{
    private string _currentTime;

    public string CurrentTime
    {
        get { return _currentTime; }
        set
        {
            _currentTime = value;
            OnPropertyChanged();
        }
    }

    public MultiComponentLabelPage()
    {
        InitializeComponent();
        BindingContext = this;
    }

    protected override void OnAppearing()
    {
        base.OnAppearing();

        Device.StartTimer(TimeSpan.FromSeconds(1), () =>
        {
            CurrentTime = DateTime.Now.ToString("hh : mm : ss");
            return true;
        });
    }
}
```

Erstellen eines benutzerdefinierten Eintragssteuerelements mit einer MaxLength-Eigenschaft

Das Xamarin Forms `Entry` Steuerelement verfügt nicht über eine `MaxLength` Eigenschaft. Um dies zu erreichen, können Sie `Entry` wie folgt `MaxLength` , indem Sie eine Bindable `MaxLength` Eigenschaft

hinzufügen. Dann müssen Sie nur das `TextChanged` Ereignis bei `Entry` abonnieren und die Länge des `Text` `TextChanged` wenn dieser aufgerufen wird:

```
class CustomEntry : Entry
{
    public CustomEntry()
    {
        base.TextChanged += Validate;
    }

    public static readonly BindableProperty MaxLengthProperty =
BindableProperty.Create(nameof(MaxLength), typeof(int), typeof(CustomEntry), 0);

    public int MaxLength
    {
        get { return (int)GetValue(MaxLengthProperty); }
        set { SetValue(MaxLengthProperty, value); }
    }

    public void Validate(object sender, TextChangedEventArgs args)
    {
        var e = sender as Entry;
        var val = e?.Text;

        if (string.IsNullOrEmpty(val))
            return;

        if (MaxLength > 0 && val.Length > MaxLength)
            val = val.Remove(val.Length - 1);

        e.Text = val;
    }
}
```

Verwendung in XAML:

```
   <ContentView xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:customControls="clr-namespace:CustomControls;assembly=CustomControls"
         x:Class="Views.TestView">
<ContentView.Content>
    <customControls:CustomEntry MaxLength="10" />
</ContentView.Content>
```

Benutzerdefinierte Steuerelemente erstellen online lesen: https://riptutorial.com/de/xamarin-forms/topic/3913/benutzerdefinierte-steuerelemente-erstellen

Kapitel 12: Benutzerdefinierte Steuerelemente erstellen

Einführung

Jede `Xamarin.Forms` Ansicht hat einen begleitenden Renderer für jede Plattform, die eine Instanz eines systemeigenen Steuerelements erstellt. Wenn eine Ansicht auf einer bestimmten Plattform `ViewRenderer` , wird die `ViewRenderer` Klasse instanziiert.

Der Vorgang dafür ist wie folgt:

Erstellen Sie ein benutzerdefiniertes Xamarin.Forms-Steuerelement.

Verwenden Sie das benutzerdefinierte Steuerelement von Xamarin.Forms.

Erstellen Sie den benutzerdefinierten Renderer für das Steuerelement auf jeder Plattform.

Examples

Ein CheckBox-Steuerelement implementieren

In diesem Beispiel implementieren wir eine benutzerdefinierte Checkbox für Android und iOS.

Erstellen des benutzerdefinierten Steuerelements

```
namespace CheckBoxCustomRendererExample
{
    public class Checkbox : View
    {
        public static readonly BindableProperty IsCheckedProperty =
BindableProperty.Create<Checkbox, bool>(p => p.IsChecked, true, propertyChanged: (s, o, n) =>
{ (s as Checkbox).OnChecked(new EventArgs()); });
        public static readonly BindableProperty ColorProperty =
BindableProperty.Create<Checkbox, Color>(p => p.Color, Color.Default);

        public bool IsChecked
        {
            get
            {
                return (bool)GetValue(IsCheckedProperty);
            }
            set
            {
                SetValue(IsCheckedProperty, value);
            }
        }

        public Color Color
        {
```

```
            get
            {
                return (Color)GetValue(ColorProperty);
            }
            set
            {
                SetValue(ColorProperty, value);
            }
        }

        public event EventHandler Checked;

        protected virtual void OnChecked(EventArgs e)
        {
            if (Checked != null)
                Checked(this, e);
        }
    }
}
```

Wir beginnen mit dem Android Custom Renderer, indem wir eine neue Klasse (`CheckboxCustomRenderer`) im `Android` Bereich unserer Lösung `CheckboxCustomRenderer` .

Ein paar wichtige Details zu beachten:

- Wir brauchen die Spitze unserer Klasse mit dem ExportRenderer Attribute zu markieren , so dass der Renderer registriert ist `Xamarin.Forms` . Auf diese Weise `Xamarin.Forms` wird diese Renderer verwenden , wenn es versucht , unser erstellen `Checkbox` Objekt auf `Android` .
- Wir arbeiten `OnElementChanged` der `OnElementChanged` Methode, bei der wir unser `OnElementChanged` instanziieren und einrichten.

Verbrauch der benutzerdefinierten Steuerung

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" xmlns:local="clr-
namespace:CheckBoxCustomRendererExample"
x:Class="CheckBoxCustomRendererExample.CheckBoxCustomRendererExamplePage">
    <StackLayout Padding="20">
        <local:Checkbox Color="Aqua" />
    </StackLayout>
</ContentPage>
```

Erstellen des benutzerdefinierten Renderers auf jeder Plattform

Zum Erstellen der benutzerdefinierten Renderer-Klasse gehen Sie wie folgt vor:

1. Erstellen Sie eine Unterklasse der `ViewRenderer<T1,T2>` Klasse `ViewRenderer<T1,T2>` , die das benutzerdefinierte Steuerelement darstellt. Das erste `CheckBox` sollte das benutzerdefinierte Steuerelement sein, für das der Renderer bestimmt ist, in diesem Fall `CheckBox` . Das zweite Typargument sollte das native Steuerelement sein, das das benutzerdefinierte

Steuerelement implementiert.
2. `OnElementChanged` die `OnElementChanged` Methode, die die benutzerdefinierte Steuerungs- und Schreiblogik darstellt, um sie anzupassen. Diese Methode wird aufgerufen, wenn das entsprechende `Xamarin.Forms` Steuerelement erstellt wird.
3. Fügen `ExportRenderer` der benutzerdefinierten Renderer-Klasse ein `ExportRenderer` Attribut hinzu, um anzugeben, dass es zum Rendern des benutzerdefinierten `Xamarin.Forms` Steuerelements verwendet wird. Dieses Attribut wird verwendet, um den benutzerdefinierten Renderer bei `Xamarin.Forms` zu registrieren.

Erstellen des benutzerdefinierten Renderers für Android

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.Droid
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBox>
    {
        private CheckBox checkBox;

        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);
            var model = e.NewElement;
            checkBox = new CheckBox(Context);
            checkBox.Tag = this;
            CheckboxPropertyChanged(model, null);
            checkBox.SetOnClickListener(new ClickListener(model));
            SetNativeControl(checkBox);
        }
        private void CheckboxPropertyChanged(Checkbox model, String propertyName)
        {
            if (propertyName == null || Checkbox.IsCheckedProperty.PropertyName ==
propertyName)
            {
                checkBox.Checked = model.IsChecked;
            }

            if (propertyName == null || Checkbox.ColorProperty.PropertyName == propertyName)
            {
                int[][] states = {
                    new int[] { Android.Resource.Attribute.StateEnabled}, // enabled
                    new int[] {Android.Resource.Attribute.StateEnabled}, // disabled
                    new int[] {Android.Resource.Attribute.StateChecked}, // unchecked
                    new int[] { Android.Resource.Attribute.StatePressed}  // pressed
                };
                var checkBoxColor = (int)model.Color.ToAndroid();
                int[] colors = {
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor,
                    checkBoxColor
                };
                var myList = new Android.Content.Res.ColorStateList(states, colors);
                checkBox.ButtonTintList = myList;

            }
        }
```

```
        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            if (checkBox != null)
            {
                base.OnElementPropertyChanged(sender, e);

                CheckboxPropertyChanged((Checkbox)sender, e.PropertyName);
            }
        }

        public class ClickListener : Java.Lang.Object, IOnClickListener
        {
            private Checkbox _myCheckbox;
            public ClickListener(Checkbox myCheckbox)
            {
                this._myCheckbox = myCheckbox;
            }
            public void OnClick(global::Android.Views.View v)
            {
                _myCheckbox.IsChecked = !_myCheckbox.IsChecked;
            }
        }
    }
}
```

Erstellen des benutzerdefinierten Renderers für iOS

Da in iOS das Kontrollkästchen nicht integriert ist, erstellen `CheckBoxView` zunächst eine `CheckBoxView` dann einen Renderer für unser Kontrollkästchen `CheckBoxView` .

Die `CheckBoxView` basiert auf zwei Bildern, nämlich checked_checkbox.png und unchecked_checkbox.png, sodass die Eigenschaft `Color` ignoriert wird.

Die CheckBox-Ansicht:

```
namespace CheckBoxCustomRendererExample.iOS
{
    [Register("CheckBoxView")]
    public class CheckBoxView : UIButton
    {
        public CheckBoxView()
        {
            Initialize();
        }

        public CheckBoxView(CGRect bounds)
            : base(bounds)
        {
            Initialize();
        }

        public string CheckedTitle
        {
            set
            {
                SetTitle(value, UIControlState.Selected);
```

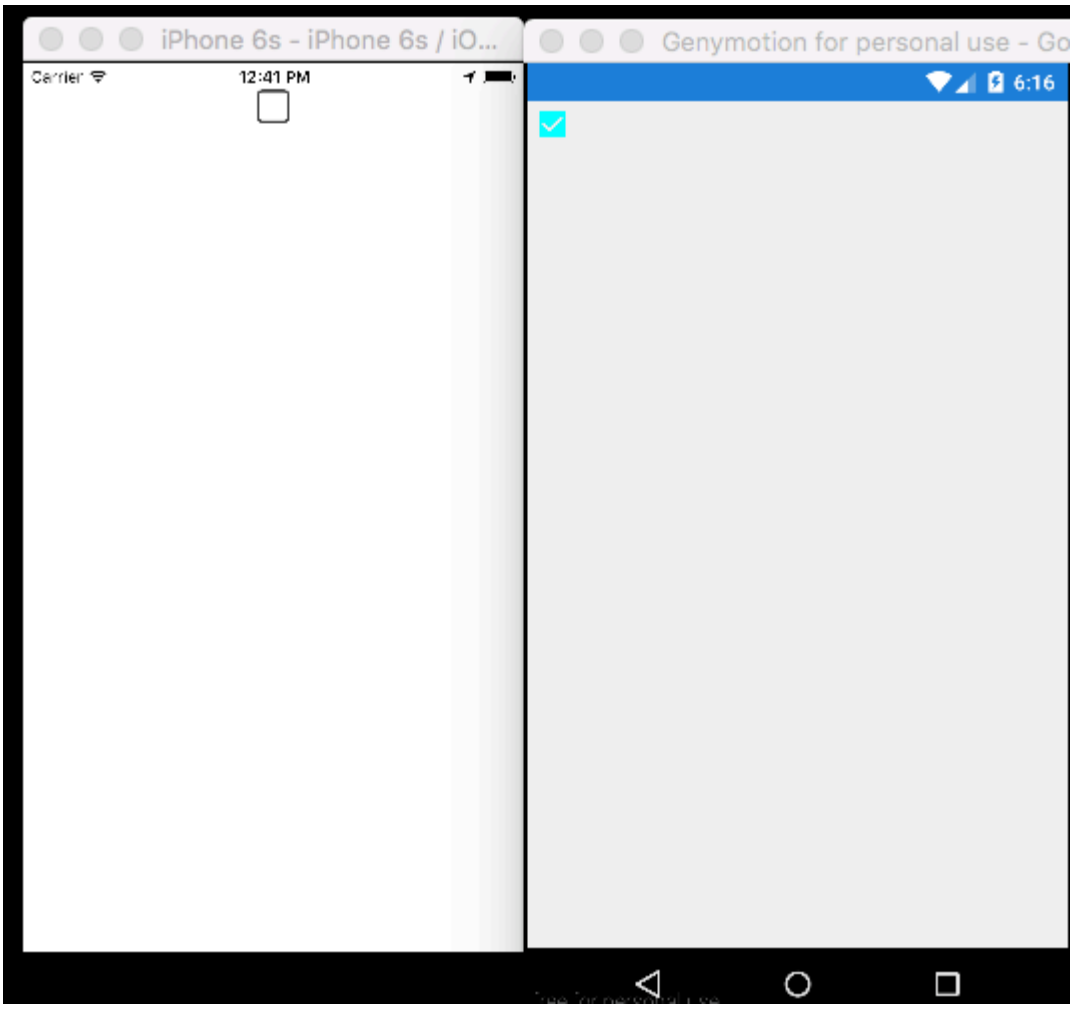
```
            }
        }

        public string UncheckedTitle
        {
            set
            {
                SetTitle(value, UIControlState.Normal);
            }
        }

        public bool Checked
        {
            set { Selected = value; }
            get { return Selected; }
        }

        void Initialize()
        {
            ApplyStyle();

            TouchUpInside += (sender, args) => Selected = !Selected;
            // set default color, because type is not UIButtonType.System
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Normal);
            SetTitleColor(UIColor.DarkTextColor, UIControlState.Selected);
        }

        void ApplyStyle()
        {
            SetImage(UIImage.FromBundle("Images/checked_checkbox.png"),
UIControlState.Selected);
            SetImage(UIImage.FromBundle("Images/unchecked_checkbox.png"),
UIControlState.Normal);
        }
    }
}
```

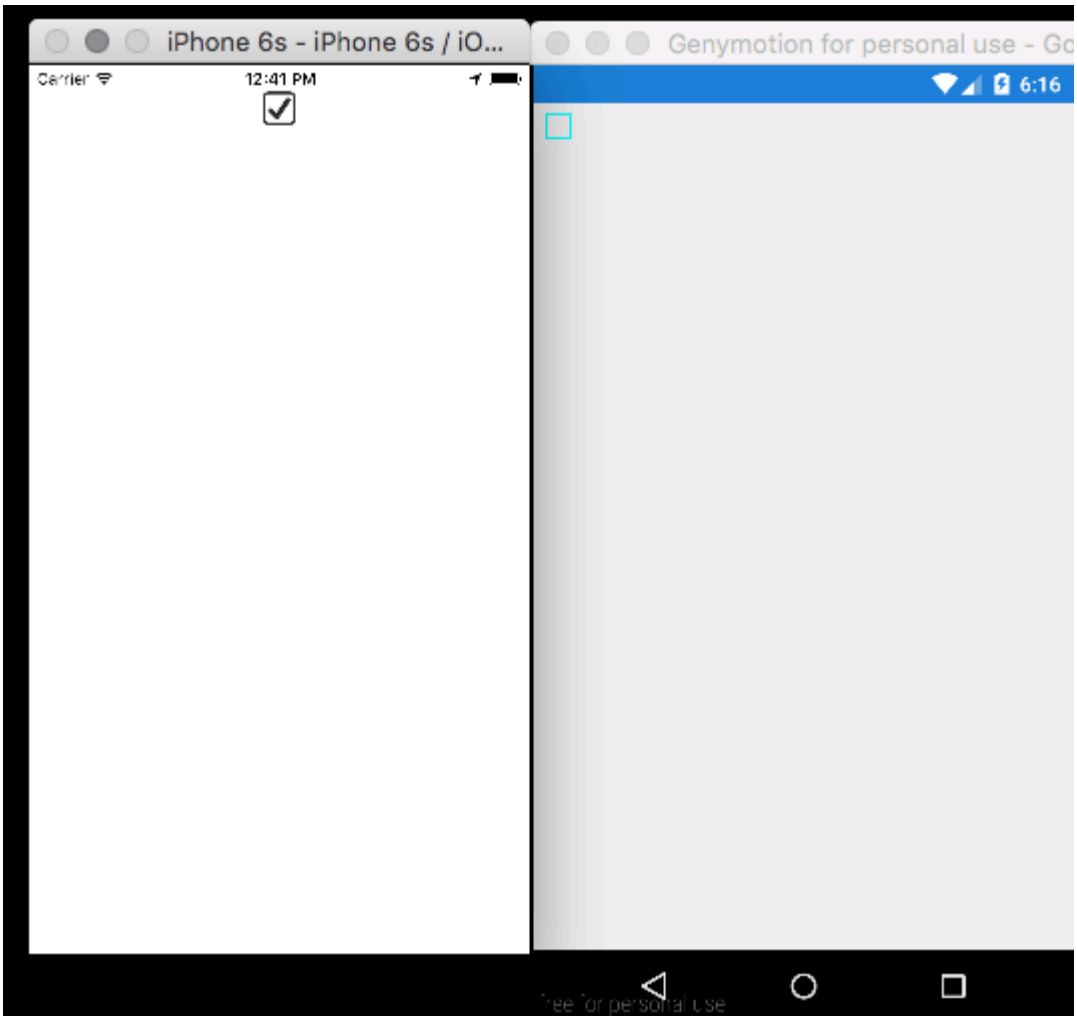
Der benutzerdefinierte CheckBox-Renderer:

```
[assembly: ExportRenderer(typeof(Checkbox), typeof(CheckBoxRenderer))]
namespace CheckBoxCustomRendererExample.iOS
{
    public class CheckBoxRenderer : ViewRenderer<Checkbox, CheckBoxView>
    {

        /// <summary>
        /// Handles the Element Changed event
        /// </summary>
        /// <param name="e">The e.</param>
        protected override void OnElementChanged(ElementChangedEventArgs<Checkbox> e)
        {
            base.OnElementChanged(e);

            if (Element == null)
                return;

            BackgroundColor = Element.BackgroundColor.ToUIColor();
            if (e.NewElement != null)
            {
                if (Control == null)
```

```
                {
                    var checkBox = new CheckBoxView(Bounds);
                    checkBox.TouchUpInside += (s, args) => Element.IsChecked =
Control.Checked;
                    SetNativeControl(checkBox);
                }
                Control.Checked = e.NewElement.IsChecked;
            }

            Control.Frame = Frame;
            Control.Bounds = Bounds;

        }

        /// <summary>
        /// Handles the <see cref="E:ElementPropertyChanged" /> event.
        /// </summary>
        /// <param name="sender">The sender.</param>
        /// <param name="e">The <see cref="PropertyChangedEventArgs"/> instance containing the
event data.</param>
        protected override void OnElementPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
            base.OnElementPropertyChanged(sender, e);

            if (e.PropertyName.Equals("Checked"))
            {
                Control.Checked = Element.IsChecked;
            }
        }
    }
}
```

Ergebnis:


iPhone 6s - iPhone 6s / iO...
Carrier
12:41 PM
Genymotion for personal use - Go
6:16



Benutzerdefinierte Steuerelemente erstellen online lesen: https://riptutorial.com/de/xamarin-forms/topic/5975/benutzerdefinierte-steuerelemente-erstellen

Kapitel 13: Benutzerdefinierte Steuerelemente erstellen

Examples

Benutzerdefinierte Schaltfläche erstellen

```
/// <summary>
/// Button with some additional options
/// </summary>
public class TurboButton : Button
{
    public static readonly BindableProperty StringDataProperty = BindableProperty.Create(
      propertyName: "StringData",
      returnType: typeof(string),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(string));

    public static readonly BindableProperty IntDataProperty = BindableProperty.Create(
      propertyName: "IntData",
      returnType: typeof(int),
      declaringType: typeof(ButtonWithStorage),
      defaultValue: default(int));

    /// <summary>
    /// You can put here some string data
    /// </summary>
    public string StringData
    {
        get { return (string)GetValue(StringDataProperty); }
        set { SetValue(StringDataProperty, value); }
    }

    /// <summary>
    /// You can put here some int data
    /// </summary>
    public int IntData
    {
        get { return (int)GetValue(IntDataProperty); }
        set { SetValue(IntDataProperty, value); }
    }

    public TurboButton()
    {
        PropertyChanged += CheckIfPropertyLoaded;
    }

    /// <summary>
    /// Called when one of properties is changed
    /// </summary>
    private void CheckIfPropertyLoaded(object sender, PropertyChangedEventArgs e)
    {
        //example of using PropertyChanged
        if(e.PropertyName == "IntData")
        {
```

```
            //IntData is now changed, you can operate on updated value
        }
    }
}
```

Verwendung in XAML:

```
<?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="SomeApp.Pages.SomeFolder.Example"
    xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp">
    <StackLayout>
        <customControls:TurboButton x:Name="exampleControl" IntData="2" StringData="Test" />
    </StackLayout>
</ContentPage>
```

Jetzt können Sie Ihre Eigenschaften in c # verwenden:

```
exampleControl.IntData
```

Beachten Sie, dass Sie selbst angeben müssen, wo sich Ihre TurboButton-Klasse in Ihrem Projekt befindet. Ich habe es in dieser Zeile gemacht:

```
xmlns:customControls="clr-namespace:SomeApp.CustomControls;assembly=SomeApp"
```

Sie können "customControls" frei in einen anderen Namen ändern. Es liegt an Ihnen, wie Sie es nennen werden.

Benutzerdefinierte Steuerelemente erstellen online lesen: https://riptutorial.com/de/xamarin-forms/topic/6592/benutzerdefinierte-steuerelemente-erstellen

Kapitel 14: Caching

Examples

Caching mit Akavache

Über Akavache

Akavache ist eine unglaublich nützliche Bibliothek, die Reichweitenfunktionen zum Zwischenspeichern Ihrer Daten bietet. Akavache bietet eine Speicherschnittstelle mit Schlüsselwerten und arbeitet auf der Oberseite von SQLite3. Sie müssen Ihr Schema nicht synchron halten, da es sich um eine No-SQL-Lösung handelt, die es ideal für die meisten mobilen Anwendungen macht, insbesondere wenn Ihre App häufig ohne Datenverlust aktualisiert werden muss.

Empfehlungen für Xamarin

Akavache ist definitiv die beste Caching-Bibliothek für Xamarin-Anwendungen, wenn Sie nicht mit stark relativen Daten, binären oder wirklich großen Datenmengen arbeiten müssen. Verwenden Sie Akavache in den folgenden Fällen:

- Sie benötigen Ihre App, um die Daten für einen bestimmten Zeitraum zwischenzuspeichern (Sie können ein Ablaufzeitlimit für jede zu speichernde Entität konfigurieren.
- Sie möchten, dass Ihre App offline arbeitet.
- Es ist schwierig, das Schema Ihrer Daten zu ermitteln und einzufrieren. Beispielsweise haben Sie Listen mit verschiedenen typisierten Objekten.
- Es genügt für Sie, über einen einfachen Schlüsselwertzugriff auf die Daten zu verfügen, und Sie müssen keine komplexen Abfragen durchführen.

Akavache ist keine "Wunderwaffe" für die Datenspeicherung. Überlegen Sie in den folgenden Fällen zweimal, wie Sie es verwenden können:

- Ihre Datenentitäten haben viele Beziehungen zueinander.
- Sie brauchen Ihre App nicht wirklich, um offline zu arbeiten.
- Sie haben eine riesige Menge an Daten, die lokal gespeichert werden müssen.
- Sie müssen Ihre Daten von Version zu Version migrieren.
- Sie müssen komplexe SQL-typische Abfragen ausführen, z. B. Gruppierung, Projektionen usw.

  Tatsächlich können Sie Ihre Daten manuell migrieren, indem Sie sie mit aktualisierten Feldern lesen und zurückschreiben.

Einfaches Beispiel

Die Interaktion mit Akavache erfolgt hauptsächlich über ein Objekt namens `BlobCache` .

Die meisten Methoden von Akavache liefern reaktive Observable, die Sie aber dank Erweiterungsmethoden auch erwarten können.

```
using System.Reactive.Linq;   // IMPORTANT - this makes await work!

// Make sure you set the application name before doing any inserts or gets
BlobCache.ApplicationName = "AkavacheExperiment";

var myToaster = new Toaster();
await BlobCache.UserAccount.InsertObject("toaster", myToaster);

//
// ...later, in another part of town...
//

// Using async/await
var toaster = await BlobCache.UserAccount.GetObject<Toaster>("toaster");

// or without async/await
Toaster toaster;

BlobCache.UserAccount.GetObject<Toaster>("toaster")
    .Subscribe(x => toaster = x, ex => Console.WriteLine("No Key!"));
```

Fehlerbehandlung

```
Toaster toaster;

try {
    toaster = await BlobCache.UserAccount.GetObjectAsync("toaster");
} catch (KeyNotFoundException ex) {
    toaster = new Toaster();
}

// Or without async/await:
toaster = await BlobCache.UserAccount.GetObjectAsync<Toaster>("toaster")
    .Catch(Observable.Return(new Toaster()));
```

Caching online lesen: https://riptutorial.com/de/xamarin-forms/topic/3644/caching

Kapitel 15: CarouselView - Vorabversion

Bemerkungen

CarouselView ist ein Xamarin-Steuerelement, das alle Arten von Ansichten enthalten kann. Diese Vorabversionskontrolle kann nur in Xamarin Forms-Projekten verwendet werden.

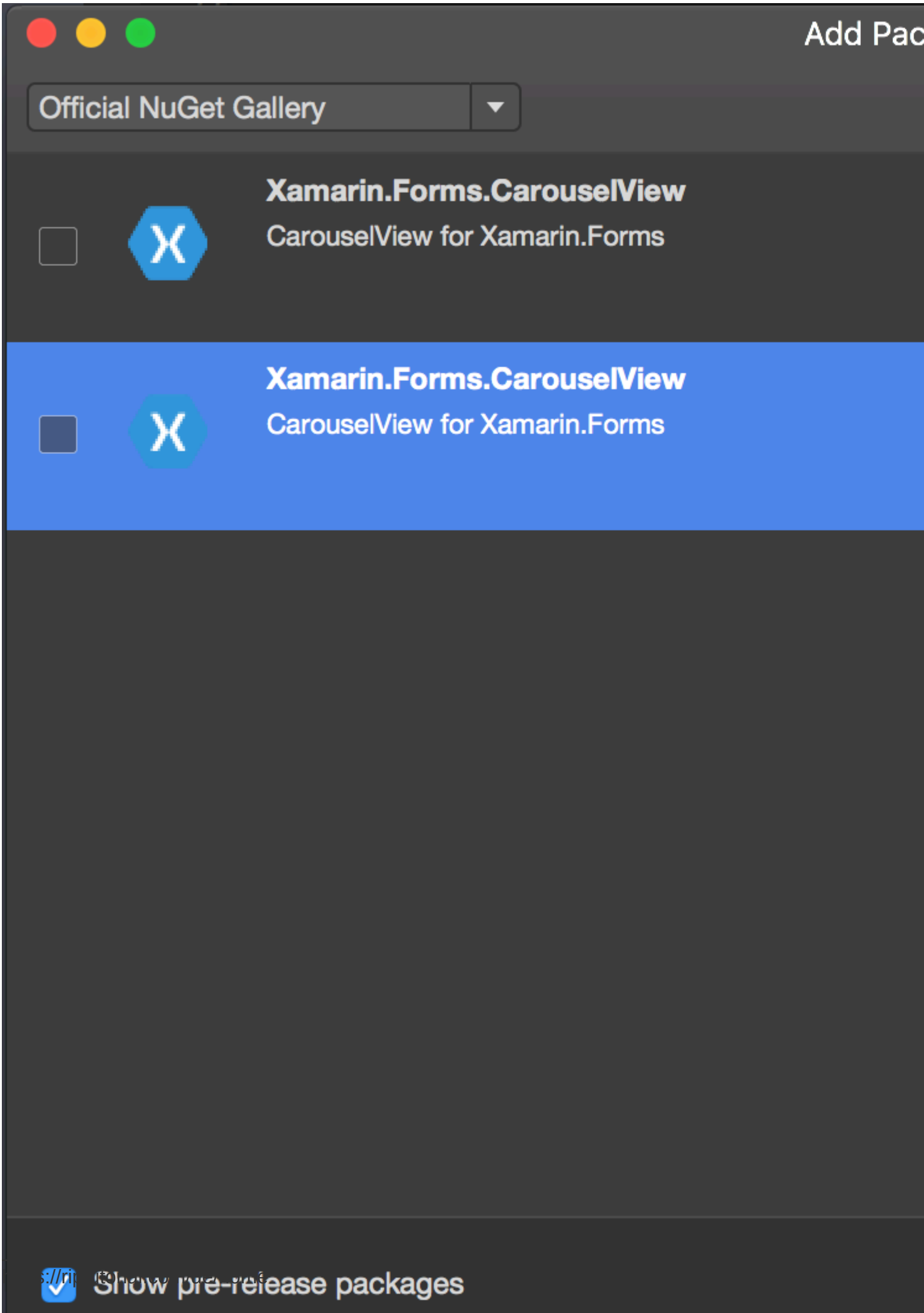
In dem von James Montemagno bereitgestellten Beispiel wird auf dem Blog von Xamarin CarouselView zum Anzeigen von Bildern verwendet.

In diesem Moment ist CarouselView nicht in Xamarin.Forms integriert. Um dies in Ihren Projekten zu verwenden, müssen Sie das NuGet-Package hinzufügen (siehe Beispiel oben).

Examples

Karussellansicht importieren

Der einfachste Weg, CarouselView zu importieren, ist die Verwendung des NuGet-Packages Manager in Xamarin / Visual Studio:


Add Pac
Official NuGet Gallery
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Xamarin.Forms.CarouselView
CarouselView for Xamarin.Forms
Show pre-release packages

https://riptutorial.com/de/xamarin-forms/topic/6094/carouselview---vorabversion

Kapitel 16: Datenbindung

Bemerkungen

Mögliche Ausnahmen

System.ArrayTypeMismatchException: Es wurde versucht, auf ein Element als Typ zuzugreifen, der mit dem Array nicht kompatibel ist.

Diese Ausnahme kann auftreten, wenn versucht wird, eine Auflistung an eine nicht bindbare Eigenschaft zu binden, wenn die XAML-Vorkompilierung aktiviert ist. Ein bekanntes Beispiel ist der Versuch, an `Picker.Items` zu binden. Siehe unten.

System.ArgumentException: Objekt vom Typ 'Xamarin.Forms.Binding' kann nicht in den Typ 'System.String' konvertiert werden.

Diese Ausnahme kann auftreten, wenn versucht wird, eine Auflistung an eine nicht bindbare Eigenschaft zu binden, wenn die XAML-Vorkompilierung deaktiviert ist. Ein bekanntes Beispiel ist der Versuch, an `Picker.Items` zu binden. Siehe unten.

Die `Picker.Items` Eigenschaft ist nicht bindbar

Dieser Code verursacht einen Fehler:

```
<!-- BAD CODE: will cause an error -->
<Picker Items="{Binding MyViewModelItems}" SelectedIndex="0" />
```

Die Ausnahme kann eine der folgenden sein:

> System.ArrayTypeMismatchException: Es wurde versucht, auf ein Element als Typ zuzugreifen, der mit dem Array nicht kompatibel ist.

oder

> System.ArgumentException: Objekt vom Typ 'Xamarin.Forms.Binding' kann nicht in den Typ 'System.String' konvertiert werden.

Die `Items` Eigenschaft ist insbesondere nicht bindbar. Zu den Lösungen zählen das Erstellen eines eigenen benutzerdefinierten Steuerelements oder das Verwenden eines Steuerelements eines

Drittanbieters, z. B. `BindablePicker` von FreshEssentials . Nach der Installation des FreshEssentials NuGet-Pakets in Ihrem Projekt ist das `BindablePicker` Steuerelement des Pakets mit einer `ItemsSource` Eigenschaft verfügbar:

```
<?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:fe="clr-namespace:FreshEssentials;assembly=FreshEssentials"
             xmlns:my="clr-namespace:MyAssembly;assembly=MyAssembly"
             x:Class="MyNamespace.MyPage">
  <ContentPage.BindingContext>
    <my:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <fe:BindablePicker ItemsSource="{Binding MyViewModelItems}" SelectedIndex="0" />
  </ContentPage.Content>
</ContentPage>
```

Examples

Grundlegende Bindung an ViewModel

EntryPage.xaml:

```
<?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:vm="clr-namespace:MyAssembly.ViewModel;assembly=MyAssembly"
             x:Class="MyAssembly.EntryPage">
  <ContentPage.BindingContext>
    <vm:MyViewModel />
  </ContentPage.BindingContext>
  <ContentPage.Content>
    <StackLayout VerticalOptions="FillAndExpand"
                 HorizontalOptions="FillAndExpand"
                 Orientation="Vertical"
                 Spacing="15">
      <Label Text="Name:" />
      <Entry Text="{Binding Name}" />
      <Label Text="Phone:" />
      <Entry Text="{Binding Phone}" />
      <Button Text="Save" Command="{Binding SaveCommand}" />
    </StackLayout>
  </ContentPage.Content>
</ContentPage>
```

MyViewModel.cs:

```
using System;
using System.ComponentModel;

namespace MyAssembly.ViewModel
{
    public class MyViewModel : INotifyPropertyChanged
    {
```

```
        private string _name = String.Empty;
        private string _phone = String.Empty;

        public string Name
        {
            get { return _name; }
            set
            {
                if (_name != value)
                {
                    _name = value;
                    OnPropertyChanged(nameof(Name));
                }
            }
        }

        public string Phone
        {
            get { return _phone; }
            set
            {
                if (_phone != value)
                {
                    _phone = value;
                    OnPropertyChanged(nameof(Phone));
                }
            }
        }

        public ICommand SaveCommand { get; private set; }

        public MyViewModel()
        {
            SaveCommand = new Command(SaveCommandExecute);
        }

        private void SaveCommandExecute()
        {

        }

        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Datenbindung online lesen: https://riptutorial.com/de/xamarin-forms/topic/3915/datenbindung

Kapitel 17: DependencyService

Bemerkungen

Wenn Sie `DependencyService` , benötigen Sie normalerweise 3 Teile:

- **Schnittstelle** - Hier werden die Funktionen definiert, die Sie abstrahieren möchten.
- **Plattformimplementierung** - Eine Klasse in jedem plattformspezifischen Projekt, die die zuvor definierte Schnittstelle implementiert.
- **Registrierung** - Jede plattformspezifische Implementierungsklasse muss über ein Metadatenattribut beim `DependencyService` registriert werden. Dadurch kann der `DependencyService` Ihre Implementierung zur Laufzeit finden.

Wenn Sie `DependencyService` Sie für jede Plattform, auf die Sie abzielen, eine Implementierung bereitstellen. Wenn keine Implementierung bereitgestellt wird, schlägt die Anwendung zur Laufzeit fehl.

Examples

Schnittstelle

Die Schnittstelle definiert das Verhalten, das Sie über den `DependencyService` verfügbar machen möchten. Ein Beispiel für die Verwendung eines `DependencyService` ist ein Text-To-Speech-Dienst. Es gibt derzeit keine Abstraktion für diese Funktion in Xamarin.Forms. Sie müssen also eigene erstellen. Beginnen Sie mit der Definition einer Schnittstelle für das Verhalten:

```
public interface ITextToSpeech
{
       void Speak (string whatToSay);
}
```

Da wir unsere Schnittstelle definieren, können wir aus unserem gemeinsam genutzten Code dagegen codieren.

Anmerkung: Klassen, die die Schnittstelle implementieren, müssen über einen parameterlosen Konstruktor verfügen, um mit dem `DependencyService` .

iOS-Implementierung

Die von Ihnen definierte Schnittstelle muss in jeder Zielplattform implementiert werden. Für iOS erfolgt dies über das `AVFoundation` Framework. Die folgende Implementierung der `ITextToSpeech` Schnittstelle behandelt das Sprechen eines bestimmten Textes in Englisch.

```
using AVFoundation;

public class TextToSpeechiOS : ITextToSpeech
```

```
{
    public TextToSpeechiOS () {}

    public void Speak (string whatToSay)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (whatToSay) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

Wenn Sie Ihre Klasse erstellt haben, müssen Sie den `DependencyService` aktivieren, um ihn zur Laufzeit zu erkennen. Dies geschieht durch Hinzufügen eines Attributs `[assembly]` über der Klassendefinition und außerhalb von Namespacedefinitionen.

```
using AVFoundation;
using DependencyServiceSample.iOS;

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechiOS))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechiOS : ITextToSpeech
...
```

Dieses Attribut registriert die Klasse beim `DependencyService` sodass es verwendet werden kann, wenn eine Instanz der `ITextToSpeech` Schnittstelle benötigt wird.

Geteilter Code

Nachdem Sie plattformspezifische Klassen erstellt und registriert haben, können Sie sie mit Ihrem freigegebenen Code verknüpfen. Die folgende Seite enthält eine Schaltfläche, die die Text-zu-Sprache-Funktion mit einem vordefinierten Satz auslöst. Es verwendet `DependencyService` , um eine plattformspezifische Implementierung von `ITextToSpeech` zur Laufzeit mithilfe der nativen SDKs `ITextToSpeech` .

```
public MainPage ()
{
    var speakButton = new Button {
        Text = "Talk to me baby!",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };

    speakButton.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Xamarin Forms likes eating cake by the
ocean.");
    };

    Content = speakButton;
```

```
}
```

Wenn Sie diese Anwendung auf einem iOS- oder Android-Gerät ausführen und auf die Schaltfläche tippen, hören Sie, dass die Anwendung den angegebenen Satz spricht.

Android-Implementierung

Die Android-spezifische Implementierung ist etwas komplexer, da Sie dazu `Java.Lang.Object` von einem nativen `Java.Lang.Object` zu erben und die `IOnInitListener` Schnittstelle zu implementieren. Für Android müssen Sie einen gültigen Android-Kontext für viele der SDK-Methoden bereitstellen, die verfügbar gemacht werden. Xamarin.Forms macht ein `Forms.Context` Objekt `Forms.Context` , das Ihnen einen Android-Kontext zur Verfügung stellt, den Sie in solchen Fällen verwenden können.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechAndroid : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech _speaker;

    public TextToSpeechAndroid () {}

    public void Speak (string whatToSay)
    {
        var ctx = Forms.Context;

        if (_speaker == null)
        {
            _speaker = new TextToSpeech (ctx, this);
        }
        else
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (whatToSay, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation

    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success))
        {
            var p = new Dictionary<string,string> ();
            _speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }

    #endregion
}
```

Wenn Sie Ihre Klasse erstellt haben, müssen Sie den `DependencyService` aktivieren, um ihn zur Laufzeit zu erkennen. Dies geschieht durch Hinzufügen eines Attributs `[assembly]` über der

Klassendefinition und außerhalb von Namespacedefinitionen.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechAndroid))]
namespace DependencyServiceSample.Droid {
    ...
```

Dieses Attribut registriert die Klasse beim `DependencyService` sodass es verwendet werden kann, wenn eine Instanz der `ITextToSpeech` Schnittstelle benötigt wird.

DependencyService online lesen: https://riptutorial.com/de/xamarin-forms/topic/2508/dependencyservice

Kapitel 18: Generischer Xamarin.Forms-App-Lebenszyklus? Plattformabhängig!

Examples

Der Lebenszyklus von Xamarin.Forms ist nicht der eigentliche App-Lebenszyklus, sondern eine plattformübergreifende Darstellung.

Schauen wir uns die systemeigenen App-Lebenszyklusmethoden für verschiedene Plattformen an.

Android.

```
//Xamarin.Forms.Platform.Android.FormsApplicationActivity lifecycle methods:
protected override void OnCreate(Bundle savedInstanceState);
protected override void OnDestroy();
protected override void OnPause();
protected override void OnRestart();
protected override void OnResume();
protected override void OnStart();
protected override void OnStop();
```

iOS.

```
//Xamarin.Forms.Platform.iOS.FormsApplicationDelegate lifecycle methods:
public override void DidEnterBackground(UIApplication uiApplication);
public override bool FinishedLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void OnActivated(UIApplication uiApplication);
public override void OnResignActivation(UIApplication uiApplication);
public override void WillEnterForeground(UIApplication uiApplication);
public override bool WillFinishLaunching(UIApplication uiApplication, NSDictionary
launchOptions);
public override void WillTerminate(UIApplication uiApplication);
```

Windows.

```
//Windows.UI.Xaml.Application lifecycle methods:
public event EventHandler<System.Object> Resuming;
public event SuspendingEventHandler Suspending;
protected virtual void OnActivated(IActivatedEventArgs args);
protected virtual void OnFileActivated(FileActivatedEventArgs args);
protected virtual void OnFileOpenPickerActivated(FileOpenPickerActivatedEventArgs args);
protected virtual void OnFileSavePickerActivated(FileSavePickerActivatedEventArgs args);
protected virtual void OnLaunched(LaunchActivatedEventArgs args);
protected virtual void OnSearchActivated(SearchActivatedEventArgs args);
protected virtual void OnShareTargetActivated(ShareTargetActivatedEventArgs args);
protected virtual void OnWindowCreated(WindowCreatedEventArgs args);

//Windows.UI.Xaml.Window lifecycle methods:
public event WindowActivatedEventHandler Activated;
```

```
public event WindowClosedEventHandler Closed;
public event WindowVisibilityChangedEventHandler VisibilityChanged;
```

Und jetzt die Lebenszyklusmethoden der **Xamarin.Forms-** App:

```
//Xamarin.Forms.Application lifecycle methods:
protected virtual void OnResume();
protected virtual void OnSleep();
protected virtual void OnStart();
```

Was Sie durch bloßes Beobachten der Listen leicht feststellen können, ist die Lebenszyklusperspektive der plattformübergreifenden App Xamarin.Forms erheblich vereinfacht. Sie erhalten allgemeine Informationen über den Status Ihrer App. In den meisten Produktionsfällen müssen Sie jedoch eine plattformabhängige Logik aufbauen.

Generischer Xamarin.Forms-App-Lebenszyklus? Plattformabhängig! online lesen: https://riptutorial.com/de/xamarin-forms/topic/8329/generischer-xamarin-forms-app-lebenszyklus--plattformabhangig-

Kapitel 19: Gesten

Examples

Machen Sie ein Bild tappbar, indem Sie einen TapGestureRecognizer hinzufügen

In `TapGestureRecognizer` sind einige Standarderkennungen verfügbar. Formulare, einer davon ist der `TapGestureRecognizer` .

Sie können sie zu praktisch jedem visuellen Element hinzufügen. Schauen Sie sich eine einfache Implementierung an, die an ein `Image` . So geht es im Code.

```
var tappedCommand = new Command(() =>
{
    //handle the tap
});

var tapGestureRecognizer = new TapGestureRecognizer { Command = tappedCommand };
image.GestureRecognizers.Add(tapGestureRecognizer);
```

Oder in XAML:

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer
                Command="{Binding TappedCommand}"
                NumberOfTapsRequired="2" />
  </Image.GestureRecognizers>
</Image>
```

Hier wird der Befehl über die Datenbindung festgelegt. Wie Sie sehen, können Sie `NumberOfTapsRequired` auch so einstellen, dass weitere Taps aktiviert werden, bevor Aktionen ausgeführt werden. Der Standardwert ist 1 Tippen.

Andere Gesten sind Pinch und Pan.

Zoomen Sie ein Bild mit der Pinch-Geste

Um ein `Image` (oder ein anderes visuelles Element) `PinchGestureRecognizer` zu machen, müssen wir einen `PinchGestureRecognizer` hinzufügen. So machen Sie es im Code:

```
var pinchGesture = new PinchGestureRecognizer();
pinchGesture.PinchUpdated += (s, e) => {
// Handle the pinch
};

image.GestureRecognizers.Add(pinchGesture);
```

Es kann aber auch von XAML aus gemacht werden:

```
<Image Source="waterfront.jpg">
  <Image.GestureRecognizers>
    <PinchGestureRecognizer PinchUpdated="OnPinchUpdated" />
  </Image.GestureRecognizers>
</Image>
```

Im begleitenden Event-Handler sollten Sie den Code zum Zoomen Ihres Bildes angeben. Natürlich können auch andere Anwendungen implementiert werden.

```
void OnPinchUpdated (object sender, PinchGestureUpdatedEventArgs e)
{
    // ... code here
}
```

Andere Gesten sind Tap und Pan.

Zeigen Sie den gesamten vergrößerten Bildinhalt mit PanGestureRecognizer an

Wenn Sie ein vergrößertes `Image` (oder einen anderen Inhalt) haben, können Sie das `Image` , um seinen gesamten Inhalt im vergrößerten Zustand anzuzeigen.

Dies kann durch die Implementierung des PanGestureRecognizers erreicht werden. Aus dem Code sieht das so aus:

```
var panGesture = new PanGestureRecognizer();
panGesture.PanUpdated += (s, e) => {
  // Handle the pan
};

image.GestureRecognizers.Add(panGesture);
```

Dies kann auch über XAML erfolgen:

```
<Image Source="MonoMonkey.jpg">
  <Image.GestureRecognizers>
    <PanGestureRecognizer PanUpdated="OnPanUpdated" />
  </Image.GestureRecognizers>
</Image>
```

Im Code-Behind-Event können Sie das Verschieben jetzt entsprechend behandeln. Verwenden Sie diese Methodensignatur, um damit umzugehen:

```
void OnPanUpdated (object sender, PanUpdatedEventArgs e)
{
  // Handle the pan
}
```

Platzieren Sie eine Nadel an der Stelle, an der der Benutzer den Bildschirm mit

MR.Gestures berührt hat

In Xamarins integrierte Gestenerkennungsfunktionen bieten nur sehr grundlegende Berührungsfunktionen. Es gibt zB keine Möglichkeit, die Position eines berührenden Fingers zu ermitteln. MR.Gestures ist eine Komponente, die 14 verschiedene Touch-Handling-Ereignisse hinzufügt. Die Position der berührenden Finger ist Teil der `EventArgs` die an alle MR.Gestures-Ereignisse übergeben werden.

Wenn Sie eine Pin an einer beliebigen Stelle auf dem Bildschirm platzieren möchten, verwenden Sie am besten ein `MR.Gestures.AbsoluteLayout` das das `Tapping` Ereignis behandelt.

```
<mr:AbsoluteLayout x:Name="MainLayout" Tapping="OnTapping">
    ...
</mr:AbsoluteLayout>
```

Wie Sie sehen, fühlt sich das `Tapping="OnTapping"` auch eher mit .NET als mit der Xamarins-Syntax mit den verschachtelten `GestureRecognizers` . Diese Syntax wurde von iOS kopiert und riecht ein wenig für .NET-Entwickler.

In Ihrem Code dahinter könnten Sie den `OnTapping` Handler folgendermaßen hinzufügen:

```
private void OnTapping(object sender, MR.Gestures.TapEventArgs e)
{
    if (e.Touches?.Length > 0)
    {
        Point touch = e.Touches[0];
        var image = new Image() { Source = "pin" };
        MainLayout.Children.Add(image, touch);
    }
}
```

Anstelle des `Tapping` Ereignisses können Sie auch den `TappingCommand` und an Ihr ViewModel binden. `TappingCommand` würde jedoch die Dinge in diesem einfachen Beispiel komplizieren.

Weitere Beispiele für MR.Gestures finden Sie in der GestureSample-App auf GitHub und auf der MR.Gestures-Website . Diese zeigen auch, wie alle anderen Berührungsereignisse mit Ereignishandlern, Befehlen, MVVM, ... verwendet werden.

Gesten online lesen: https://riptutorial.com/de/xamarin-forms/topic/3914/gesten

Kapitel 20: Kontaktauswahl - Xamarin-Formulare (Android und iOS)

Bemerkungen

Contact Picker XF (Android und iOS)

Examples

contact_picker.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
    public class App : Application
    {
        public App ()
        {
            // The root page of your application
            MainPage = new MyPage();
        }

        protected override void OnStart ()
        {
            // Handle when your app starts
        }

        protected override void OnSleep ()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume ()
        {
            // Handle when your app resumes
        }
    }
}
```

MyPage.cs

```
using System;

using Xamarin.Forms;

namespace contact_picker
{
    public class MyPage : ContentPage
```

```
    {
        Button button;
        public MyPage ()
        {
            button = new Button {
                Text = "choose contact"
            };

            button.Clicked += async (object sender, EventArgs e) => {

                    if (Device.OS == TargetPlatform.iOS) {
                        await Navigation.PushModalAsync (new ChooseContactPage ());
                        }
                    else if (Device.OS == TargetPlatform.Android)
                    {
                        MessagingCenter.Send (this, "android_choose_contact", "number1");
                    }


            };

            Content = new StackLayout {
                Children = {
                    new Label { Text = "Hello ContentPage" },
                    button
                }
            };
        }

        protected override void OnSizeAllocated (double width, double height)
        {
            base.OnSizeAllocated (width, height);

            MessagingCenter.Subscribe<MyPage, string> (this, "num_select", (sender, arg) => {
                DisplayAlert ("contact", arg, "OK");
            });

        }
    }
}
```

Wählen SieContactPicker.cs

```
using System;
using Xamarin.Forms;

namespace contact_picker
{
    public class ChooseContactPage : ContentPage
    {
        public ChooseContactPage ()
        {

        }
    }
}
```

Wählen SieContactActivity.cs

```
using Android.App;
using Android.OS;
using Android.Content;
using Android.Database;
using Xamarin.Forms;

namespace contact_picker.Droid
{

    [Activity (Label = "ChooseContactActivity")]

    public class ChooseContactActivity : Activity
    {
        public string type_number = "";
        protected override void OnCreate (Bundle savedInstanceState)
        {

            base.OnCreate (savedInstanceState);

            Intent intent = new Intent(Intent.ActionPick,
Android.Provider.ContactsContract.CommonDataKinds.Phone.ContentUri);
            StartActivityForResult(intent, 1);
        }

        protected override void OnActivityResult (int requestCode, Result resultCode, Intent
data)
        {
            // TODO Auto-generated method stub

            base.OnActivityResult (requestCode, resultCode, data);
            if (requestCode == 1) {
                if (resultCode == Result.Ok) {

                    Android.Net.Uri contactData = data.Data;
                    ICursor cursor = ContentResolver.Query(contactData, null, null, null,
null);

                    cursor.MoveToFirst();

                    string number =
cursor.GetString(cursor.GetColumnIndexOrThrow(Android.Provider.ContactsContract.CommonDataKinds.Phone.N



                    var twopage_renderer = new MyPage();
                    MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select",
number);
                    Finish ();
                    Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();


                }
                else if (resultCode == Result.Canceled)
                {
                    Finish ();
                }
            }
        }
    }
}
```

MainActivity.cs

```
using System;

using Android.App;
using Android.Content;
using Android.Content.PM;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Android.OS;
using Xamarin.Forms;

namespace contact_picker.Droid
{
    [Activity (Label = "contact_picker.Droid", Icon = "@drawable/icon", MainLauncher = true,
ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
    public class MainActivity :
global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
    {
        protected override void OnCreate (Bundle bundle)
        {
            base.OnCreate (bundle);

            global::Xamarin.Forms.Forms.Init (this, bundle);

            LoadApplication (new App ());


            MessagingCenter.Subscribe<MyPage, string>(this, "android_choose_contact", (sender,
args) => {
                Intent i = new Intent (Android.App.Application.Context,
typeof(ChooseContactActivity));
                i.PutExtra ("number1", args);
                StartActivity (i);
            });
        }
    }
}
```

Wählen SieContactRenderer.cs

```
using UIKit;
using AddressBookUI;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;
using contact_picker;
using contact_picker.iOS;


[assembly: ExportRenderer (typeof(ChooseContactPage), typeof(ChooseContactRenderer))]

namespace contact_picker.iOS
{
    public partial class ChooseContactRenderer : PageRenderer
    {
        ABPeoplePickerNavigationController _contactController;
```

```
        public string type_number;

        protected override void OnElementChanged (VisualElementChangedEventArgs e)
        {
            base.OnElementChanged (e);

            var page = e.NewElement as ChooseContactPage;

            if (e.OldElement != null || Element == null) {
                return;

            }

        }

        public override void ViewDidLoad ()
        {
            base.ViewDidLoad ();

            _contactController = new ABPeoplePickerNavigationController ();

            this.PresentModalViewController (_contactController, true); //display contact
chooser


            _contactController.Cancelled += delegate {
                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();

                this.DismissModalViewController (true); };

            _contactController.SelectPerson2 += delegate(object sender,
ABPeoplePickerSelectPerson2EventArgs e) {

                var getphones = e.Person.GetPhones();
                string number = "";

                if (getphones == null)
                {
                    number = "Nothing";
                }
                else if (getphones.Count > 1)
                {
                    //il ya plus de 2 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value + "/" + number;
                    }
                }
                else if (getphones.Count == 1)
                {
                    //il ya 1 num de telephone
                    foreach(var t in getphones)
                    {
                        number = t.Value;
                    }
                }


                Xamarin.Forms.Application.Current.MainPage.Navigation.PopModalAsync ();
```

```
                var twopage_renderer = new MyPage();
                MessagingCenter.Send<MyPage, string> (twopage_renderer, "num_select", number);
                this.DismissModalViewController (true);



            };
        }

        public override void ViewDidUnload ()
        {
            base.ViewDidUnload ();

            // Clear any references to subviews of the main view in order to
            // allow the Garbage Collector to collect them sooner.
            //
            // e.g. myOutlet.Dispose (); myOutlet = null;

            this.DismissModalViewController (true);
        }

        public override bool ShouldAutorotateToInterfaceOrientation (UIInterfaceOrientation
toInterfaceOrientation)
        {
            // Return true for supported orientations
            return (toInterfaceOrientation != UIInterfaceOrientation.PortraitUpsideDown);
        }
    }
}
```

Kontaktauswahl - Xamarin-Formulare (Android und iOS) online lesen: https://riptutorial.com/de/xamarin-forms/topic/6659/kontaktauswahl---xamarin-formulare--android-und-ios-

Kapitel 21: ListViews verwenden

Einführung

In dieser Dokumentation wird beschrieben, wie Sie die verschiedenen Komponenten von Xamarin Forms ListView verwenden

Examples

Zum Aktualisieren in XAML und hinterem Code ziehen

Um die `ListView` Pull in einer `ListView` in Xamarin zu `ListView` , müssen Sie zunächst angeben, dass `PullToRefresh` aktiviert ist, und dann den Namen des Befehls angeben, den Sie beim `ListView` der `ListView` aufrufen `ListView` :

```
<ListView x:Name="itemListView" IsPullToRefreshEnabled="True" RefreshCommand="Refresh">
```

Das Gleiche kann im Code dahinter erreicht werden:

```
itemListView.IsPullToRefreshEnabled = true;
itemListView.RefreshCommand = Refresh;
```

Dann müssen Sie angeben, was der `Refresh` in Ihrem Code bewirkt:

```
public ICommand Refresh
{
    get
    {
        itemListView.IsRefreshing = true; //This turns on the activity
                                          //Indicator for the ListView
        //Then add your code to execute when the ListView is pulled
        itemListView.IsRefreshing = false;
    }
}
```

ListViews verwenden online lesen: https://riptutorial.com/de/xamarin-forms/topic/9487/listviews-verwenden

Kapitel 22: MessagingCenter

Einführung

Xamarin.Forms verfügt über einen integrierten Messaging-Mechanismus, um entkoppelten Code zu fördern. Auf diese Weise müssen sich Ansichtsmodelle und andere Komponenten nicht kennen. Sie können durch einen einfachen Messaging-Vertrag kommunizieren.

Grundsätzlich gibt es zwei Hauptzutaten für die Verwendung des `MessagingCenter` .

Abonnieren Nachrichten mit einer bestimmten Signatur (dem Vertrag) abhören und Code ausführen, wenn eine Nachricht empfangen wird. Eine Nachricht kann mehrere Abonnenten haben.

Senden ; Senden einer Nachricht an die Abonnenten, auf die sie reagieren soll.

Examples

Einfaches Beispiel

Hier sehen Sie ein einfaches Beispiel für die Verwendung des MessagingCenters in Xamarin.Forms.

Zuerst werfen wir einen Blick auf das Abonnieren einer Nachricht. Im `FooMessaging` Modell abonnieren wir eine Nachricht von der `MainPage` . Die Nachricht sollte "Hi" sein und wenn wir sie erhalten, registrieren wir einen Handler, der die Eigenschaft `Greeting` festlegt. Schließlich bedeutet `this` dass sich die aktuelle `FooMessaging` Instanz für diese Nachricht registriert.

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage> (this, "Hi", (sender) => {
            this.Greeting = "Hi there!";
        });
    }
}
```

Um eine Nachricht senden zu können, die diese Funktionalität auslöst, benötigen wir eine Seite namens `MainPage` und implementieren Code wie darunter.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage> (this, "Hi");
    }
```

```
}
```

In unserer `MainPage` wir eine Schaltfläche mit einem Handler, der eine Nachricht sendet. `this` sollte eine Instanz von `MainPage` .

Argumente übergeben

Sie können auch Argumente mit einer Nachricht übergeben, mit der Sie arbeiten möchten.

Wir werden die Klassifizierung aus unserem vorherigen Beispiel verwenden und sie erweitern. Fügen Sie im Empfangsteil direkt hinter dem Aufruf der `Subscribe` Methode den Typ des erwarteten Arguments hinzu. Stellen Sie außerdem sicher, dass Sie auch die Argumente in der Handler-Signatur deklarieren.

```
public class FooMessaging
{
    public string Greeting { get; set; }

    public FooMessaging()
    {
        MessagingCenter.Subscribe<MainPage, string> (this, "Hi", (sender, arg) => {
            this.Greeting = arg;
        });
    }
}
```

Stellen Sie beim Senden einer Nachricht sicher, dass Sie den Argumentwert angeben. Außerdem fügen Sie hier den Typ direkt hinter der `Send` Methode und den Argumentwert hinzu.

```
public class MainPage : Page
{
    private void OnButtonClick(object sender, EventArgs args)
    {
        MessagingCenter.Send<MainPage, string> (this, "Hi", "Hi there!");
    }
}
```

In diesem Beispiel wird eine einfache Zeichenfolge verwendet. Sie können jedoch auch beliebige andere (komplexe) Objekte verwenden.

Abbestellen

Wenn Sie keine Nachrichten mehr erhalten müssen, können Sie ihn einfach abbestellen. Sie können es so machen:

```
MessagingCenter.Unsubscribe<MainPage> (this, "Hi");
```

Wenn Sie Argumente angeben, müssen Sie die vollständige Signatur wie folgt abbestellen:

```
MessagingCenter.Unsubscribe<MainPage, string> (this, "Hi");
```

MessagingCenter online lesen: https://riptutorial.com/de/xamarin-

forms/topic/9672/messagingcenter

Kapitel 23: Mit Karten arbeiten

Bemerkungen

Wenn Sie Ihr Projekt auf einem anderen Computer ausführen, müssen Sie dafür einen neuen API-Schlüssel generieren, da SHA-1-Fingerabdrücke nicht für verschiedene Build-Maschinen passen.

Sie können das Projekt erkunden, wie im Beispiel *Hinzufügen einer Karte in Xamarin.Forms* hier beschrieben

Examples

Hinzufügen einer Karte in Xamarin.Forms (Xamarin Studio)

Sie können die nativen Karten-APIs auf jeder Plattform einfach mit Xamarin Forms verwenden. Sie müssen lediglich das *Xamarin.Forms.Maps*- Paket von nuget herunterladen und in jedem Projekt (einschließlich des PCL-Projekts) installieren.

Karteninitialisierung

Zunächst müssen Sie diesen Code zu Ihren plattformspezifischen Projekten hinzufügen. Dazu müssen Sie den Methodenaufruf `Xamarin.FormsMaps.Init` hinzufügen, wie in den folgenden Beispielen.

---

iOS-Projekt

Datei AppDelegate.cs

```
[Register("AppDelegate")]
public partial class AppDelegate : Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
{
    public override bool FinishedLaunching(UIApplication app, NSDictionary options)
    {
        Xamarin.Forms.Forms.Init();
        Xamarin.FormsMaps.Init();

        LoadApplication(new App());

        return base.FinishedLaunching(app, options);
    }
}
```

Android-Projekt

Datei MainActivity.cs

```
[Activity(Label = "MapExample.Droid", Icon = "@drawable/icon", Theme = "@style/MyTheme",
MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize |
ConfigChanges.Orientation)]
public class MainActivity : Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle bundle)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(bundle);

        Xamarin.Forms.Forms.Init(this, bundle);
        Xamarin.FormsMaps.Init(this, bundle);

        LoadApplication(new App());
    }
}
```

Plattformkonfiguration

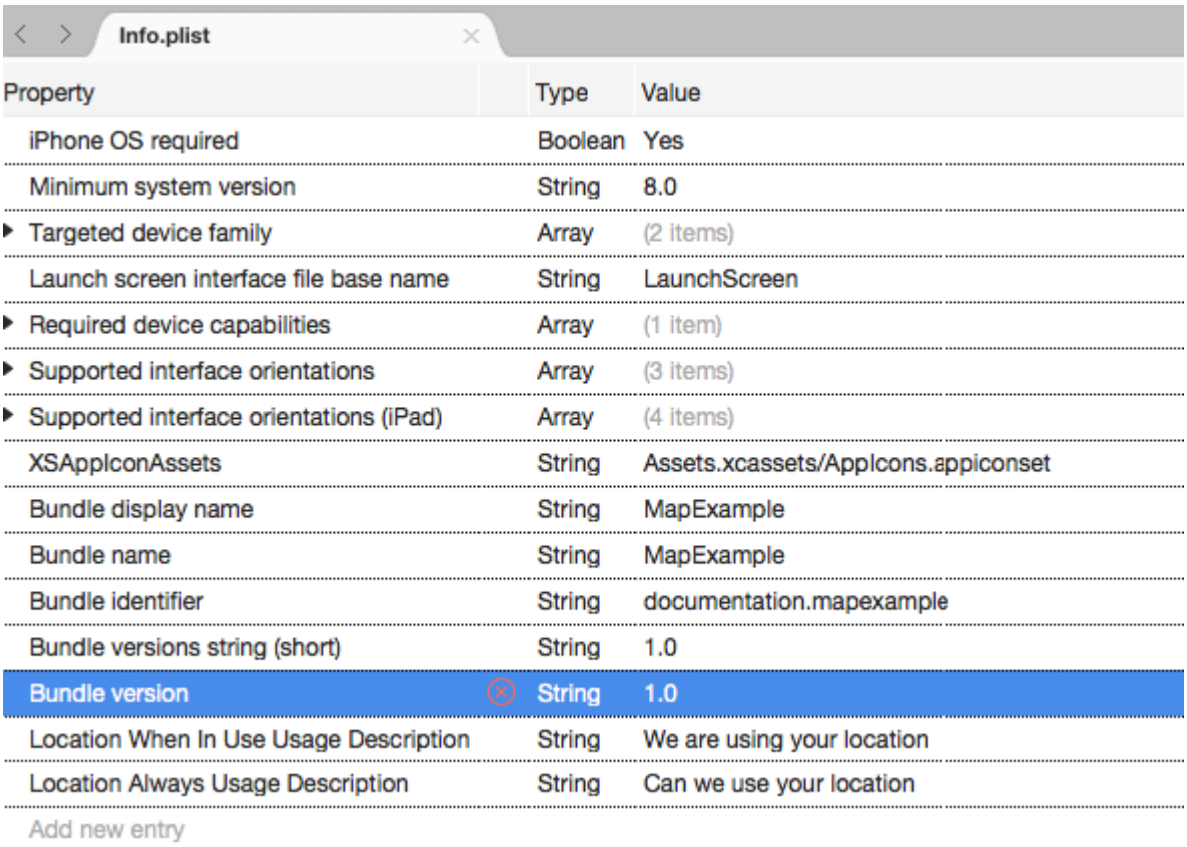
Auf einigen Plattformen sind zusätzliche Konfigurationsschritte erforderlich, bevor die Karte angezeigt wird.

---

iOS-Projekt

In iOS-Projekten müssen Sie nur 2 Einträge zu Ihrer *Info.plist*- Datei *hinzufügen* :

- `NSLocationWhenInUseUsageDescription` *Zeichenfolge* mit Wert `We are using your location`
- `NSLocationAlwaysUsageDescription` *Zeichenfolge* mit Wert `Can we use your location NSLocationAlwaysUsageDescription`



Android-Projekt

Um Google Maps verwenden zu können, müssen Sie einen API-Schlüssel generieren und ihn Ihrem Projekt hinzufügen. Folgen Sie den Anweisungen unten, um diesen Schlüssel zu erhalten:

1. (Optional) Suchen Sie nach dem Speicherort des Keytool-Tools (Standard ist `/System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands`).

2. (Optional) Öffnen Sie das Terminal und wechseln Sie zu Ihrem Keytool-Speicherort:

   ```
   cd /System/Library/Frameworks/JavaVM.framework/Versions/Current/Commands
   ```

3. Führen Sie den folgenden keytool-Befehl aus:

   ```
   keytool -list -v -keystore "/Users/[USERNAME]/.local/share/Xamarin/Mono for
   Android/debug.keystore" -alias androiddebugkey -storepass android -keypass android
   ```

   Dabei ist [USERNAME] natürlich Ihr aktueller Benutzerordner. Sie sollten etwas Ähnliches in der Ausgabe erhalten:

   ```
   Alias name: androiddebugkey
   Creation date: Jun 30, 2016
   Entry type: PrivateKeyEntry
   Certificate chain length: 1
   Certificate[1]:
   Owner: CN=Android Debug, O=Android, C=US
   Issuer: CN=Android Debug, O=Android, C=US
   Serial number: 4b5ac934
   ```

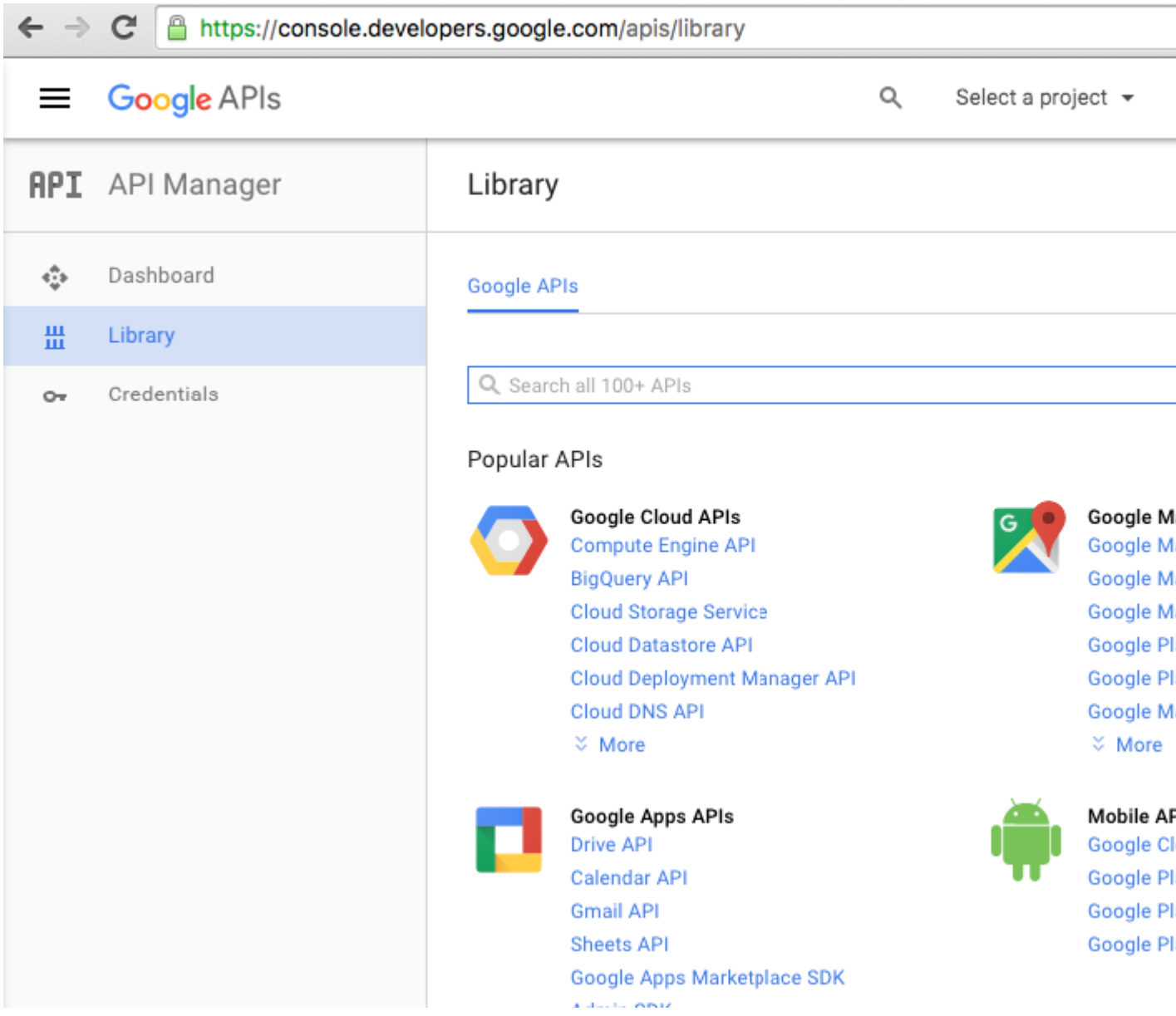
```
Valid from: Thu Jun 30 10:22:00 EEST 2016 until: Sat Jun 23 10:22:00 EEST 2046
Certificate fingerprints:
         MD5:  4E:49:A7:14:99:D6:AB:9F:AA:C7:07:E2:6A:1A:1D:CA
         SHA1: 57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
         SHA256:
70:E1:F3:5B:95:69:36:4A:82:A9:62:F3:67:B6:73:A4:DD:92:95:51:44:E3:4C:3D:9E:ED:99:03:09:9F:90:3F

         Signature algorithm name: SHA256withRSA
         Version: 3
```

4. In dieser Ausgabe benötigen wir lediglich den Fingerabdruck des SHA1-Zertifikats. In unserem Fall ist dies gleich:

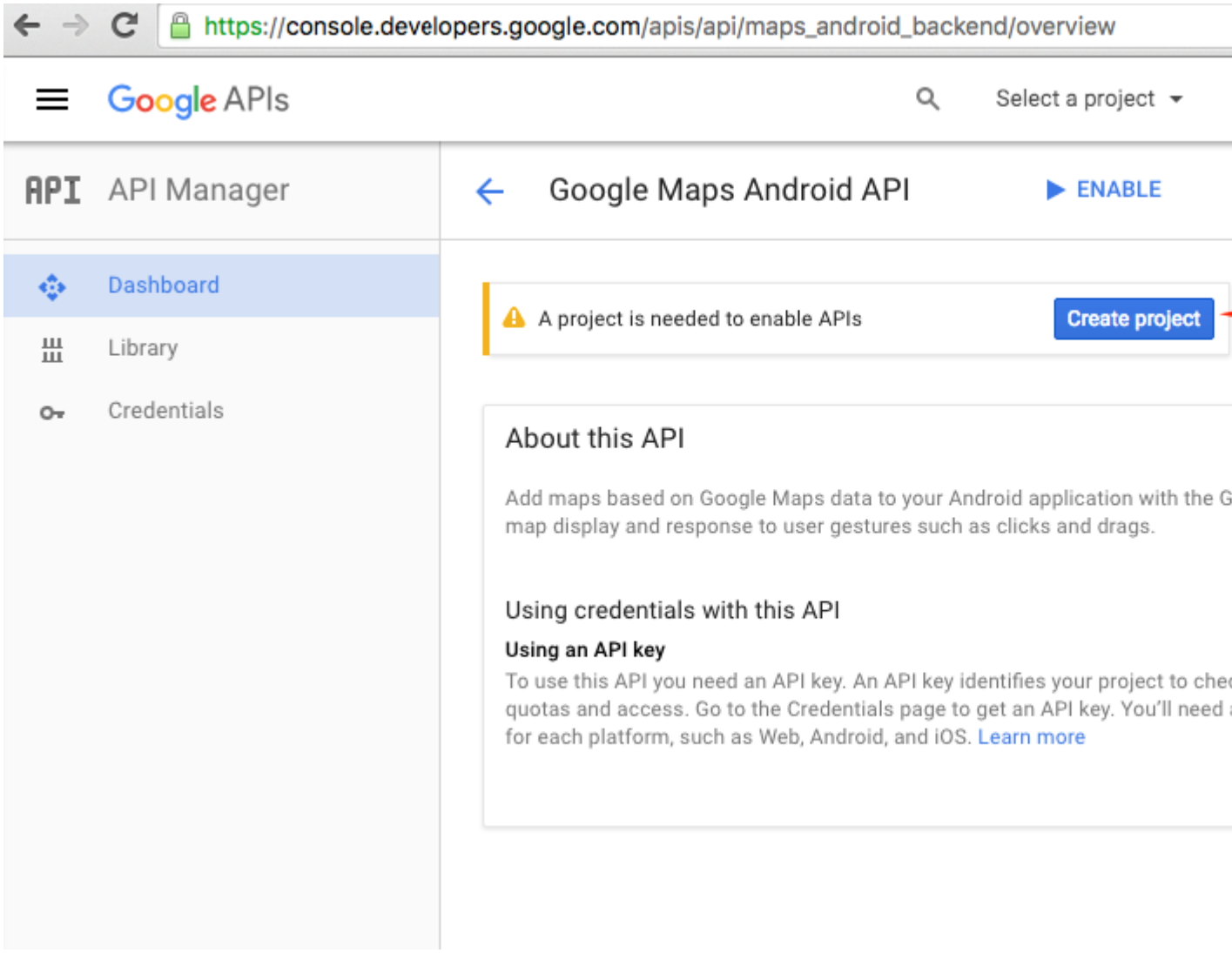
```
57:A1:E5:23:CE:49:2F:17:8D:8A:EA:87:65:44:C1:DD:1C:DA:51:95
```

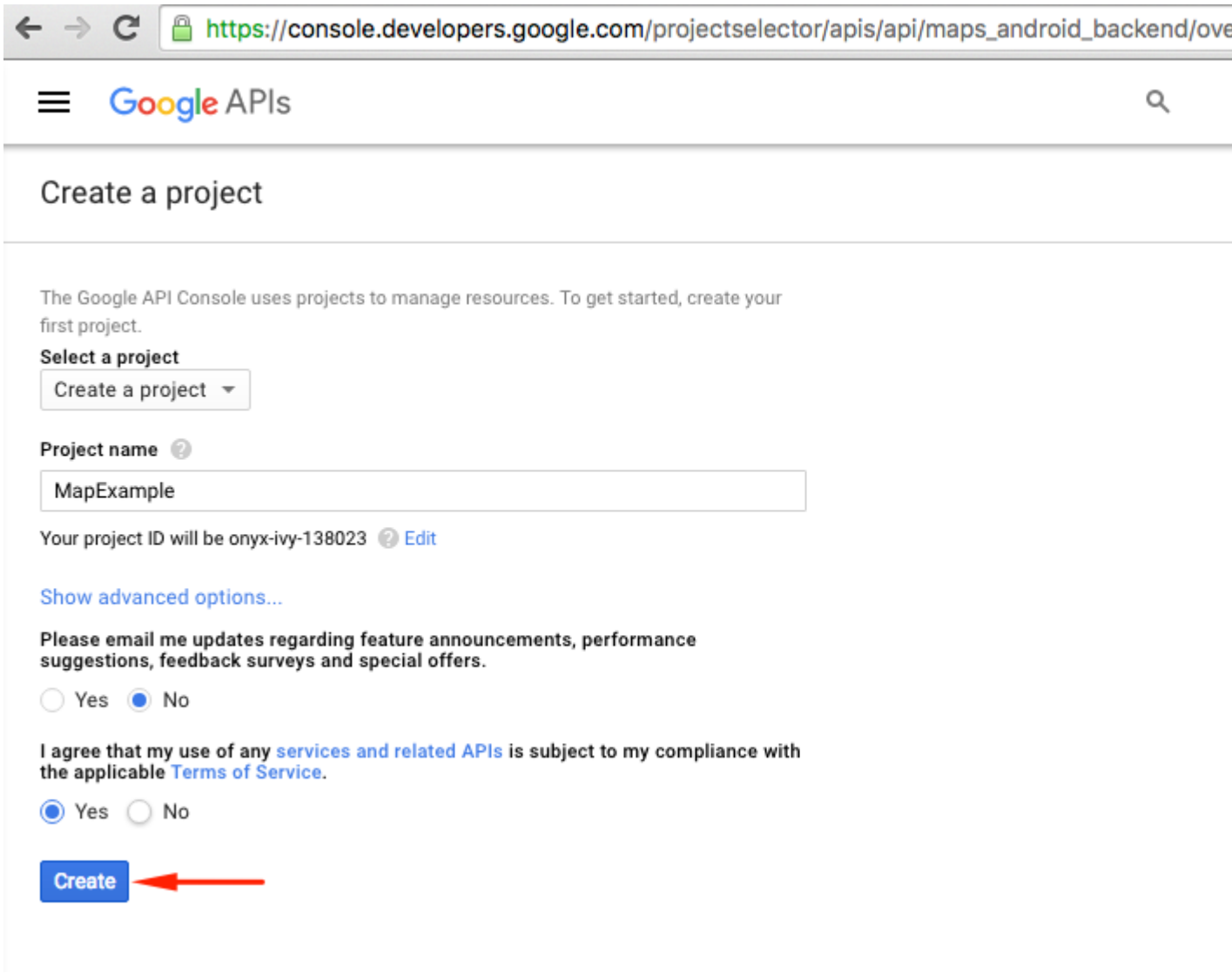
   Kopieren oder speichern Sie irgendwo diesen Schlüssel. Wir werden es später brauchen.

5. Gehen Sie zur Google Developers Console . In unserem Fall müssen Sie die Google Maps-Android-API hinzufügen. Wählen Sie sie also aus:

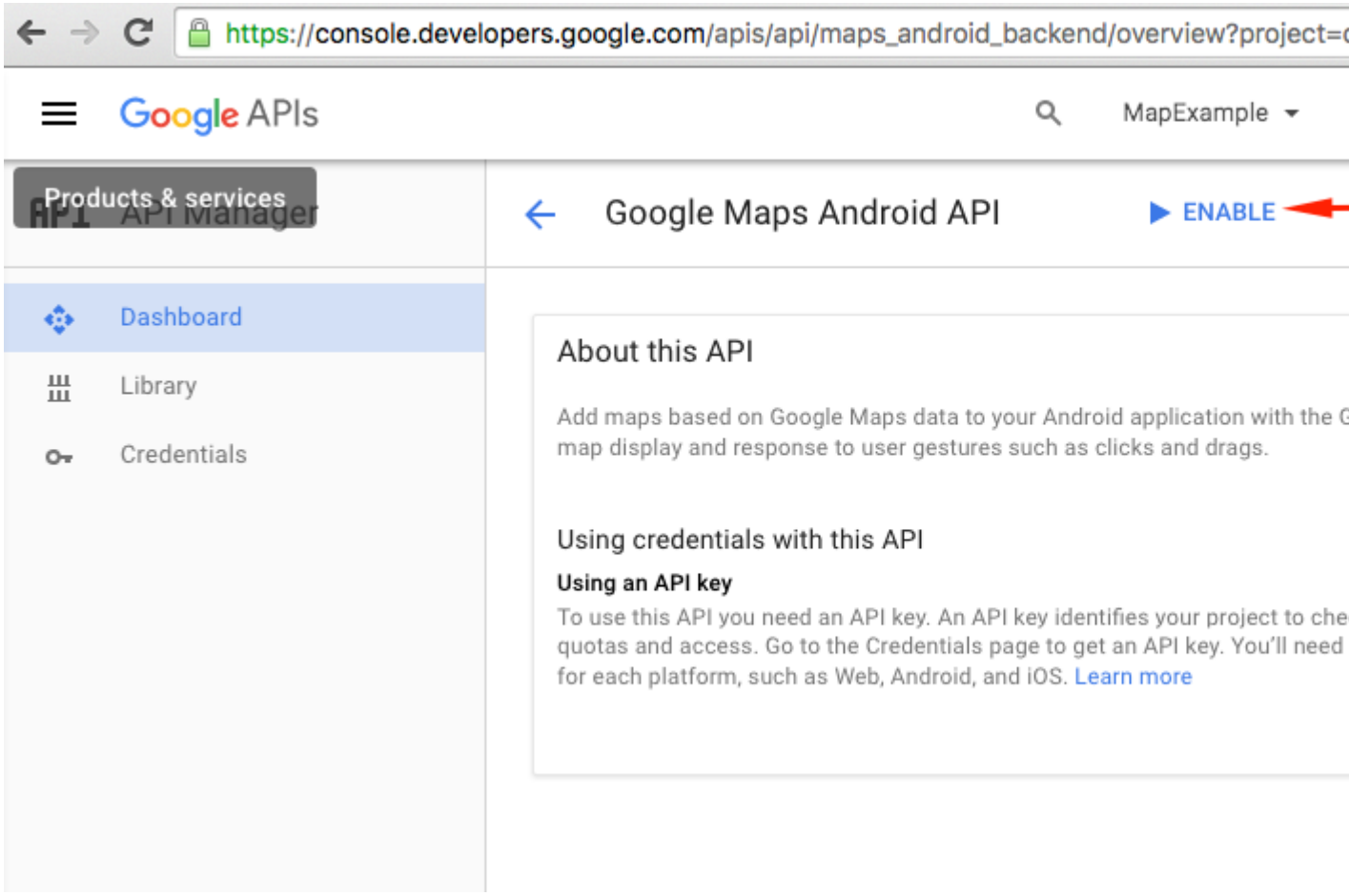


6. Google fordert Sie auf, ein Projekt zu erstellen, um APIs zu aktivieren. Folgen Sie diesem Tipp und erstellen Sie das Projekt:

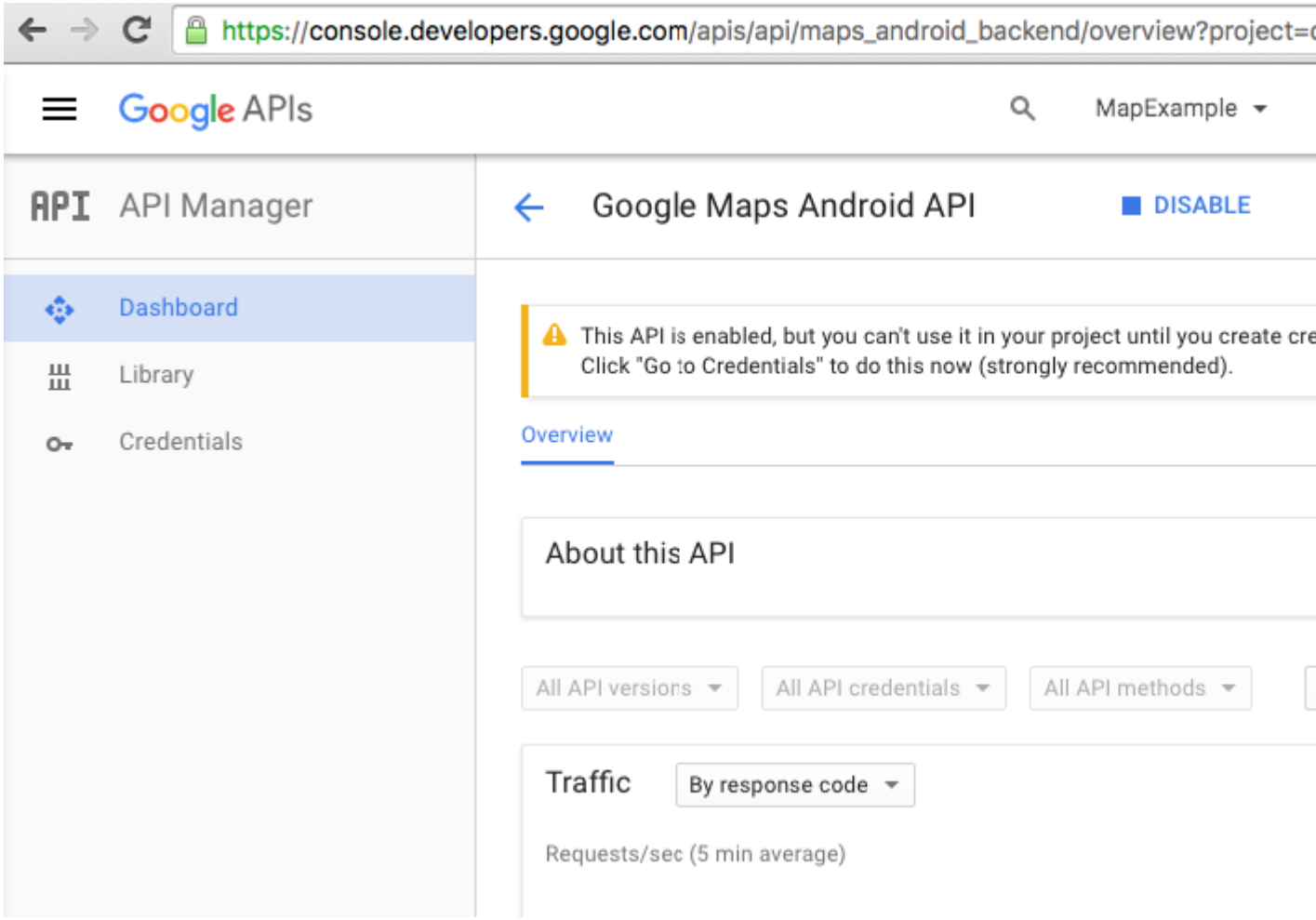




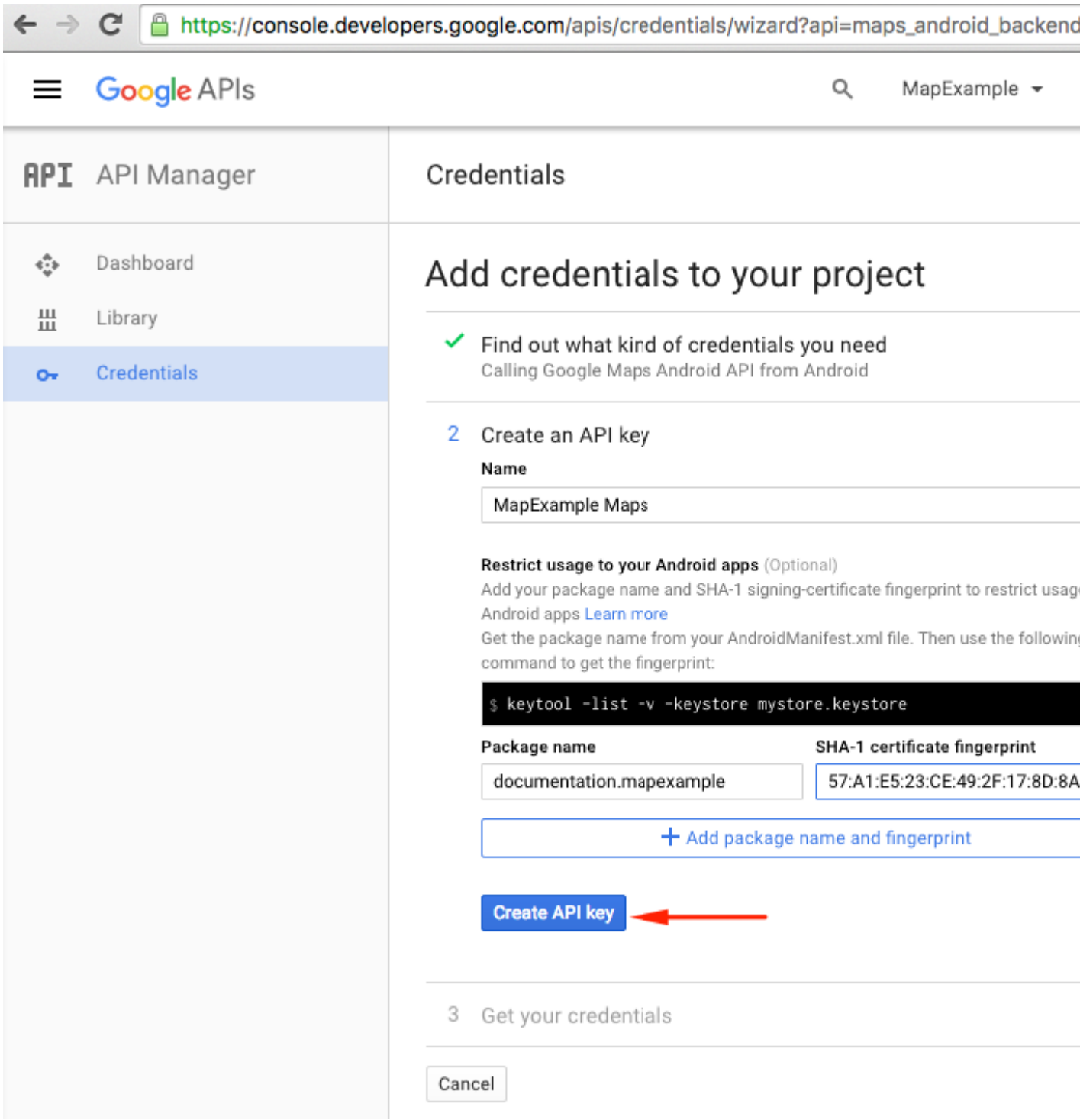
7. Aktivieren Sie die Google Maps-API für Ihr Projekt:



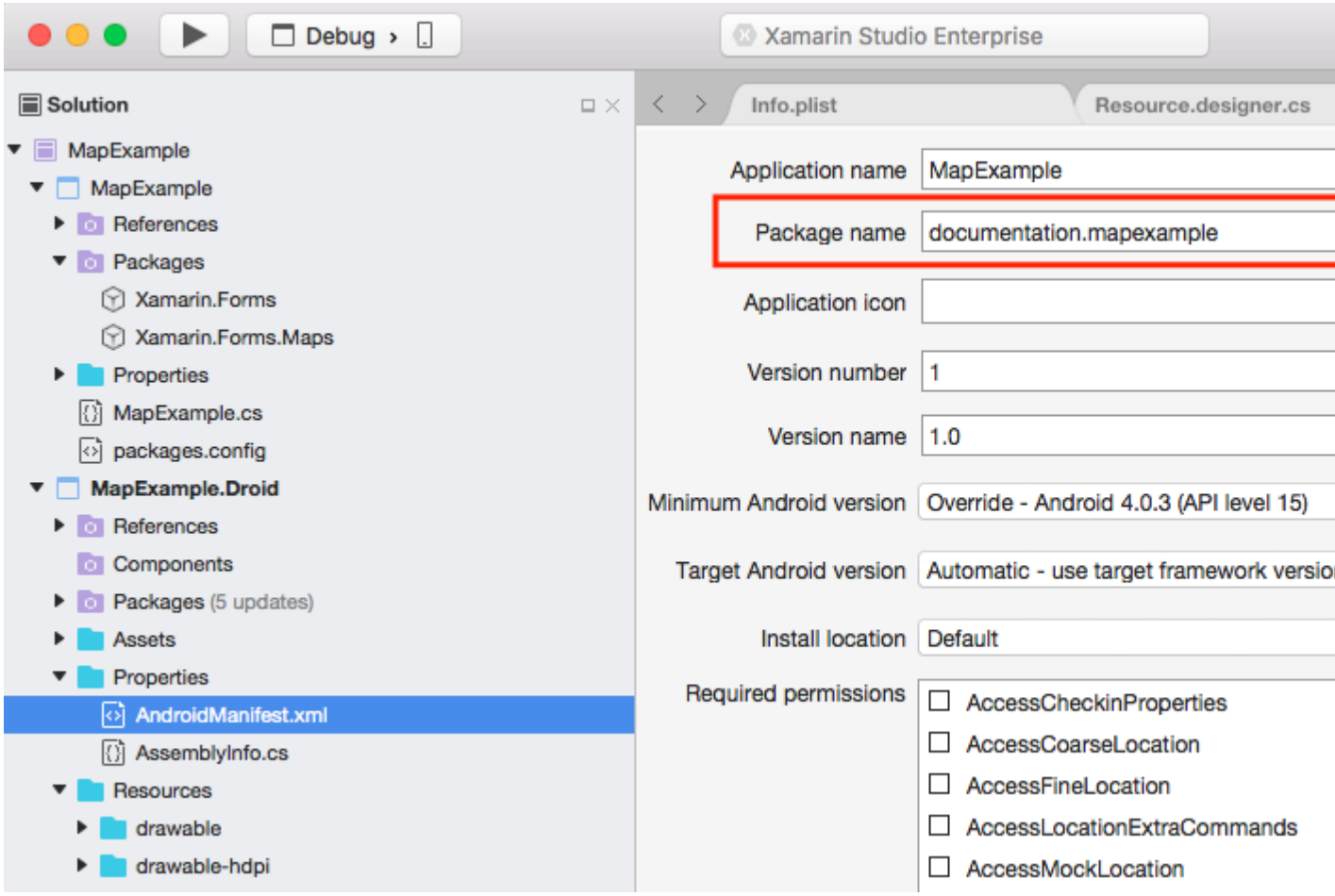
Nachdem Sie API aktiviert haben, müssen Sie Anmeldeinformationen für Ihre App erstellen. Folgen Sie diesem Tipp:



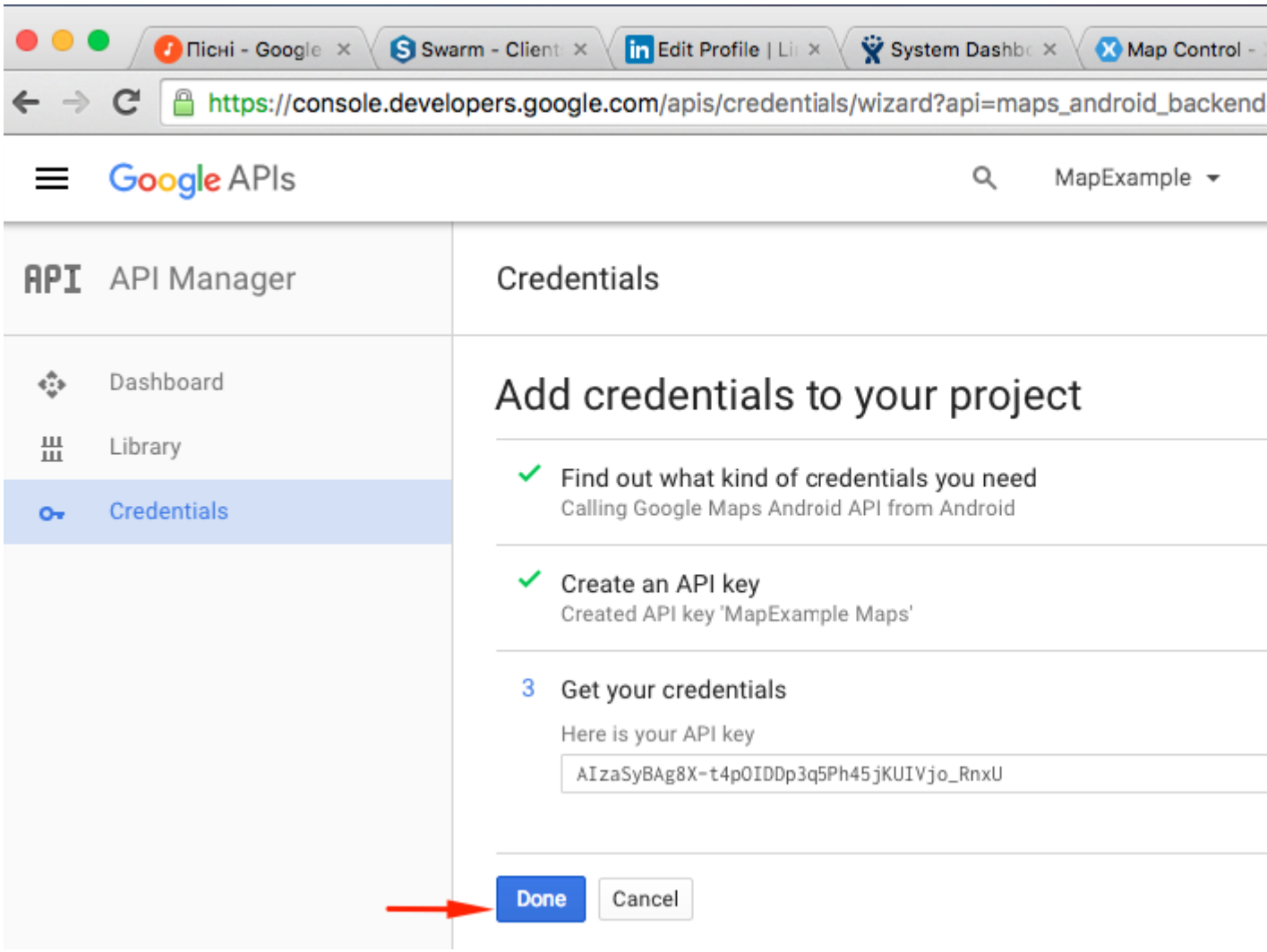
8. Wählen Sie auf der nächsten Seite die Android-Plattform aus und tippen Sie auf "Welche Anmeldeinformationen benötige ich?" , erstellen Sie einen Namen für Ihren API-Schlüssel, tippen Sie auf "Paketnamen und Fingerabdruck hinzufügen", geben Sie Ihren Paketnamen und Ihren SHA1-Fingerabdruck aus Schritt 4 ein und erstellen Sie schließlich einen API-Schlüssel:



Um Ihren Paketnamen in Xamarin Studio zu finden, gehen Sie zu Ihrer .Droid-Lösung -> AndroidManifest.xml:



9. Kopieren Sie nach der Erstellung den neuen API-Schlüssel (vergessen Sie nicht, die Schaltfläche "Fertig" zu drücken) und fügen Sie ihn in Ihre `AndroidManifest.xml` Datei ein:



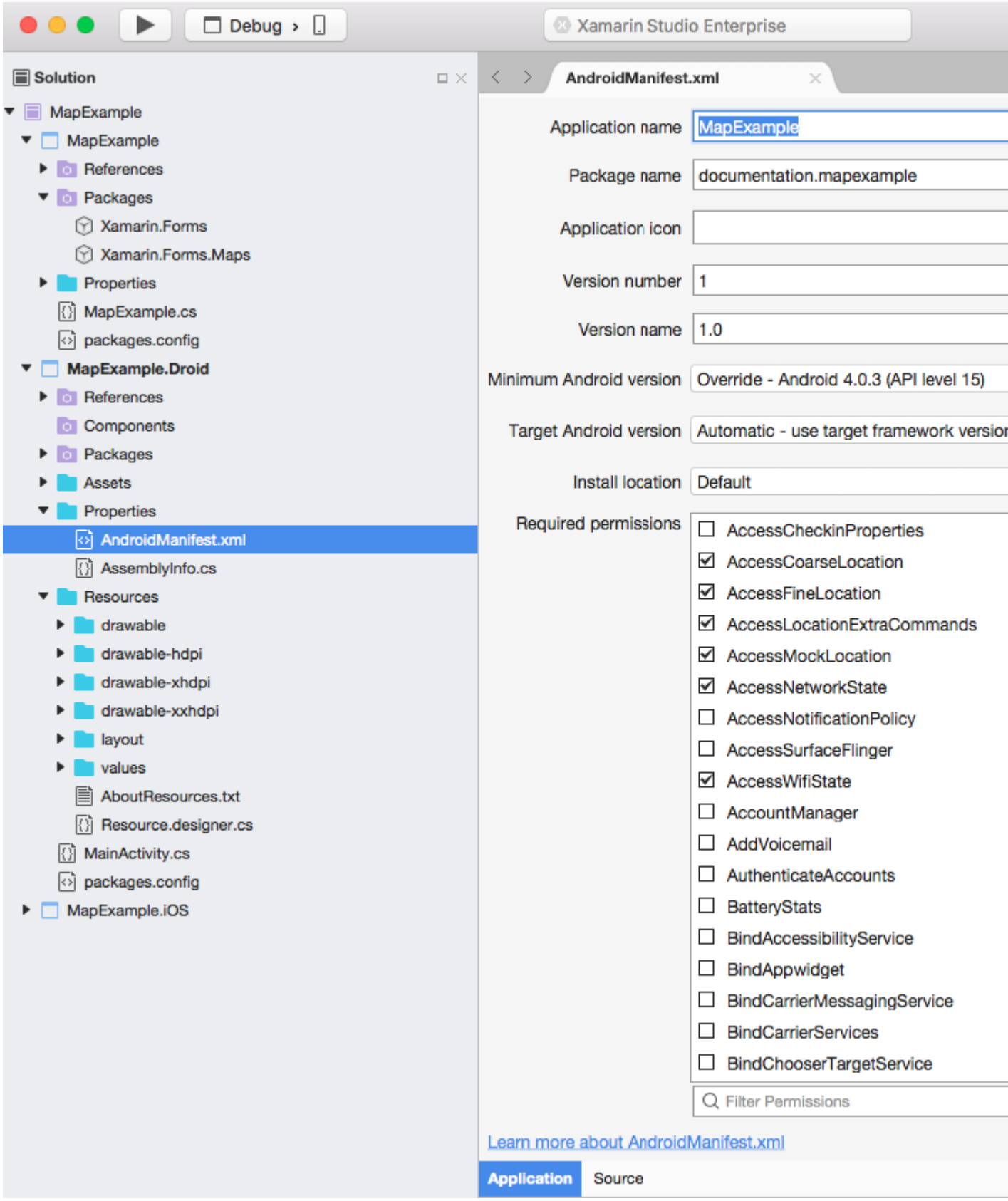
Datei AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="documentation.mapexample">
    <uses-sdk
        android:minSdkVersion="15" />
    <application
        android:label="MapExample">
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyBAg8X-t4pOIDDp3q5Ph45jKUIVjo_RnxU" />
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
    </application>
</manifest>
```

Sie müssen außerdem einige Berechtigungen in Ihrem Manifest aktivieren, um einige zusätzliche Funktionen zu aktivieren:

- Zugriff auf groben Standort
- Zugriff auf die Fine Location

- Auf zusätzliche Befehle für den Standort zugreifen
- Zugriff auf Mock Location
- Zugriff auf den Netzwerkstatus
- Zugang Wifi State
- Internet



Die letzten beiden Berechtigungen sind jedoch erforderlich, um Maps-Daten

herunterzuladen. Lesen Sie über Android-Berechtigungen , um mehr zu erfahren. Das sind alle Schritte für die Android-Konfiguration.

> *Hinweis* : Wenn Sie Ihre App auf einem Android-Simulator ausführen möchten, müssen Sie die Google Play-Dienste darauf installieren. Befolgen Sie dieses Tutorial , um Play Services auf dem Xamarin Android Player zu installieren. Wenn Sie nach der Play Store-Installation kein Update für Google Play-Dienste finden, können Sie es direkt von Ihrer App aus aktualisieren, wo Sie von Kartendiensten abhängig sind

---

Hinzufügen einer Karte

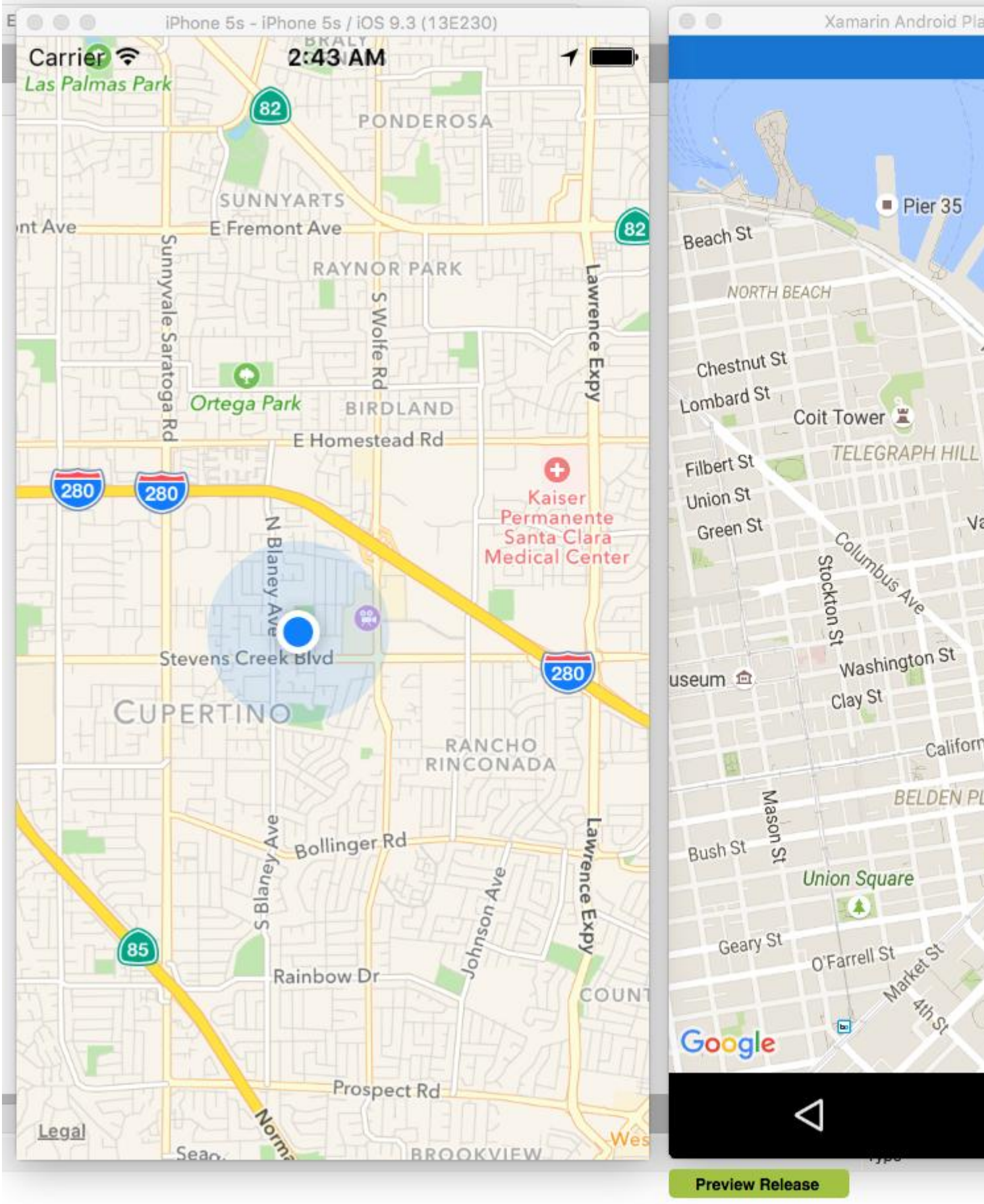
Das Hinzufügen einer Kartenansicht zu Ihrem Crossplatform-Projekt ist recht einfach. Hier ist ein Beispiel, wie das geht (Ich verwende ein PCL-Projekt ohne XAML).

PCL-Projekt

Datei MapExample.cs

```
public class App : Application
{
    public App()
    {
        var map = new Map();
        map.IsShowingUser = true;

        var rootPage = new ContentPage();
        rootPage.Content = map;

        MainPage = rootPage;
    }
}
```

Das ist alles. Wenn Sie Ihre App jetzt auf iOS oder Android ausführen, wird die Kartenansicht angezeigt:



Mit Karten arbeiten online lesen: https://riptutorial.com/de/xamarin-forms/topic/3917/mit-karten-arbeiten

Kapitel 24: Mit lokalen Datenbanken arbeiten

Examples

Verwenden von SQLite.NET in einem freigegebenen Projekt

SQLite.NET ist eine Open Source-Bibliothek, die es ermöglicht, mithilfe von `SQLite` Version 3 Unterstützung für lokale Datenbanken in einem `Xamarin.Forms` Projekt hinzuzufügen.

Die folgenden Schritte zeigen, wie diese Komponente in ein `Xamarin.Forms` Shared Project `Xamarin.Forms` :

1. Laden Sie die neueste Version der SQLite.cs- Klasse herunter und fügen Sie sie dem freigegebenen Projekt hinzu.

2. Jede Tabelle, die in die Datenbank aufgenommen werden soll, muss im Shared Project als Klasse modelliert werden. Eine Tabelle wird durch Hinzufügen von mindestens zwei Attributen in der Klasse definiert: `Table` (für die Klasse) und `PrimaryKey` (für eine Eigenschaft).

In diesem Beispiel wird dem Shared Project eine neue Klasse namens `Song` hinzugefügt, die wie folgt definiert ist:

```
using System;
using SQLite;

namespace SongsApp
{
    [Table("Song")]
    public class Song
    {
        [PrimaryKey]
        public string ID { get; set; }
        public string SongName { get; set; }
        public string SingerName { get; set; }
    }
}
```

3. `SQLiteConnection` als Nächstes eine neue Klasse mit dem Namen `Database` , die von der `SQLiteConnection` Klasse (in SQLite.cs enthalten) erbt. In dieser neuen Klasse wird der Code für den Datenbankzugriff, die Tabellenerstellung und CRUD-Operationen für jede Tabelle definiert. Beispielcode wird unten gezeigt:

```
using System;
using System.Linq;
using System.Collections.Generic;
using SQLite;

namespace SongsApp
{
    public class BaseDatos : SQLiteConnection
    {
```

```
        public BaseDatos(string path) : base(path)
        {
            Initialize();
        }

        void Initialize()
        {
            CreateTable<Song>();
        }

        public List<Song> GetSongs()
        {
            return Table<Song>().ToList();
        }

        public Song GetSong(string id)
        {
            return Table<Song>().Where(t => t.ID == id).First();
        }

        public bool AddSong(Song song)
        {
            Insert(song);
        }

        public bool UpdateSong(Song song)
        {
            Update(song);
        }

        public void DeleteSong(Song song)
        {
            Delete(song);
        }
    }
}
```

4. Wie Sie im vorherigen Schritt, der Konstruktor unserer sehen konnten `Database` - Klasse enthält einen `path` der repräsentiert den Ort der Datei , die speichert die SQLite - Datenbank - Datei. Ein statisches `Database` kann in `App.cs` deklariert `App.cs` . Der `path` ist plattformspezifisch:

```
public class App : Application
{
    public static Database DB;

    public App ()
    {
        string dbFile = "SongsDB.db3";

#if __ANDROID__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        var dbPath = System.IO.Path.Combine(docPath, dbFile);
#else
#if __IOS__
        string docPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal);
        string libPath = System.IO.Path.Combine(docPath, "..", "Library");
        var dbPath = System.IO.Path.Combine(libPath, dbFile);
#else
```

```
        var dbPath = System.IO.Path.Combine(ApplicationData.Current.LocalFolder.Path, dbFile);
#endif
#endif

        DB = new Database(dbPath);

        // The root page of your application
        MainPage = new SongsPage();
    }
}
```

5. Rufen Sie das `DB` Objekt jetzt einfach über die `App` Klasse auf, wenn Sie eine CRUD-Operation für die `Songs` Tabelle ausführen müssen. Um beispielsweise einen neuen `Song` einzufügen, nachdem der Benutzer auf eine Schaltfläche geklickt hat, können Sie den folgenden Code verwenden:

```
void AddNewSongButton_Click(object sender, EventArgs a)
{
    Song s = new Song();
    s.ID = Guid.NewGuid().ToString();
    s.SongName = songNameEntry.Text;
    s.SingerName = singerNameEntry.Text;

    App.DB.AddSong(song);
}
```

Arbeiten mit lokalen Datenbanken mit xamarin.forms in Visual Studio 2015

SQlite-Beispiel Schritt für Schritt Erklärung

1. Die folgenden Schritte zeigen, wie Sie diese Komponente in ein Xamarin.Forms Shared Project einbinden: Hinzufügen von Paketen in (pcl, Andriod, Windows, Ios) Hinzufügen von Referenzen Klicken Sie auf **Nuget-Pakete verwalten** -> klicken Sie auf Durchsuchen, um **SQLite.Net.Core-** zu installieren **PCL** , **SQLite Net Extensions** nach Abschluss der Installation überprüfen Sie es einmal in den Referenzen

2. Um die Klasse **Employee.cs** unter dem Code hinzuzufügen

```
using SQLite.Net.Attributes;

    namespace DatabaseEmployeeCreation.SqlLite
    {
        public   class Employee
        {
            [PrimaryKey,AutoIncrement]
            public int Eid { get; set; }
            public string Ename { get; set; }
            public string Address { get; set; }
            public string phonenumber { get; set; }
            public string email { get; set; }
        }
    }
```

3. Eine Schnittstelle hinzufügen ISQLite

```
using SQLite.Net;
            //using SQLite.Net;
            namespace DatabaseEmployeeCreation.SqlLite.ViewModel
            {
                public interface ISQLite
                {
                    SQLiteConnection GetConnection();
                }
            }
```

4. Erstellen Sie eine Klasse für Datenbanklogiken und Methoden unter dem folgenden Code.

mit SQLite.Net; using System.Collections.Generic; using System.Linq; mit Xamarin.Forms; Namespace DatabaseEmployeeCreation.SqlLite.ViewModel {öffentliche Klasse DatabaseLogic {statisches Objekt locker = new object (); SQLiteConnection-Datenbank;

```
    public DatabaseLogic()
    {
        database = DependencyService.Get<ISQLite>().GetConnection();
        // create the tables
        database.CreateTable<Employee>();
    }

    public IEnumerable<Employee> GetItems()
    {
        lock (locker)
        {
            return (from i in database.Table<Employee>() select i).ToList();
        }
    }

    public IEnumerable<Employee> GetItemsNotDone()
    {
        lock (locker)
        {
            return database.Query<Employee>("SELECT * FROM [Employee]");
        }
    }

    public Employee GetItem(int id)
    {
        lock (locker)
        {
            return database.Table<Employee>().FirstOrDefault(x => x.Eid == id);
        }
    }

    public int SaveItem(Employee item)
    {
        lock (locker)
        {
            if (item.Eid != 0)
            {
                database.Update(item);
                return item.Eid;
            }
            else
            {
                return database.Insert(item);
```

```
            }
        }
    }

    public int DeleteItem(int Eid)
    {
        lock (locker)
        {
            return database.Delete<Employee>(Eid);
        }
    }
}
```

}

5. Erstellen Sie eine xaml.forms EmployeeRegistration.xaml

```
    <?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="DatabaseEmployeeCreation.SqlLite.EmployeeRegistration"
      Title="{Binding Name}" >
      <StackLayout VerticalOptions="StartAndExpand" Padding="20">

        <Label Text="Ename" />
        <Entry x:Name="nameEntry" Text="{Binding Ename}"/>
        <Label Text="Address" />
        <Editor x:Name="AddressEntry" Text="{Binding Address}"/>
        <Label Text="phonenumber" />
        <Entry x:Name="phonenumberEntry" Text="{Binding phonenumber}"/>
        <Label Text="email" />
        <Entry x:Name="emailEntry" Text="{Binding email}"/>

        <Button Text="Add" Clicked="addClicked"/>

       <!-- <Button Text="Delete" Clicked="deleteClicked"/>-->

        <Button Text="Details" Clicked="DetailsClicked"/>

        <!--  <Button Text="Edit" Clicked="speakClicked"/>-->

      </StackLayout>
    </ContentPage>
```

EmployeeRegistration.cs

```
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using DatabaseEmployeeCreation.SqlLite.Views;
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Text;
    using System.Threading.Tasks;

    using Xamarin.Forms;

    namespace DatabaseEmployeeCreation.SqlLite
    {
```

```
public partial class EmployeeRegistration : ContentPage
{
    private int empid;
    private Employee obj;

    public EmployeeRegistration()
    {
        InitializeComponent();

    }

    public EmployeeRegistration(Employee obj)
    {
        this.obj = obj;
        var eid = obj.Eid;
        Navigation.PushModalAsync(new EmployeeRegistration());
        var Address = obj.Address;
        var email = obj.email;
        var Ename = obj.Ename;
        var phonenumber = obj.phonenumber;
        AddressEntry. = Address;
        emailEntry.Text = email;
        nameEntry.Text = Ename;

        //AddressEntry.Text = obj.Address;
        //emailEntry.Text = obj.email;
        //nameEntry.Text = obj.Ename;
        //phonenumberEntry.Text = obj.phonenumber;

        Employee empupdate = new Employee(); //updateing Values
        empupdate.Address = AddressEntry.Text;
        empupdate.Ename = nameEntry.Text;
        empupdate.email = emailEntry.Text;
        empupdate.Eid = obj.Eid;
        App.Database.SaveItem(empupdate);
        Navigation.PushModalAsync(new EmployeeRegistration());

    }

    public EmployeeRegistration(int empid)
    {
        this.empid = empid;
        Employee lst = App.Database.GetItem(empid);
        //var Address = lst.Address;
        //var email = lst.email;
        //var Ename = lst.Ename;
        //var phonenumber = lst.phonenumber;
        //AddressEntry.Text = Address;
        //emailEntry.Text = email;
        //nameEntry.Text = Ename;
        //phonenumberEntry.Text = phonenumber;

        // to retriva values based on id to
        AddressEntry.Text = lst.Address;
        emailEntry.Text = lst.email;
        nameEntry.Text = lst.Ename;
        phonenumberEntry.Text = lst.phonenumber;

        Employee empupdate = new Employee(); //updateing Values
        empupdate.Address = AddressEntry.Text;
        empupdate.email = emailEntry.Text;
```

```
                App.Database.SaveItem(empupdate);
                Navigation.PushModalAsync(new EmployeeRegistration());
            }

            void addClicked(object sender, EventArgs e)
            {
                //var createEmp = (Employee)BindingContext;
                Employee emp = new Employee();
                emp.Address = AddressEntry.Text;
                emp.email = emailEntry.Text;
                emp.Ename = nameEntry.Text;
                emp.phonenumber = phonenumberEntry.Text;
                App.Database.SaveItem(emp);
                this.Navigation.PushAsync(new EmployeeDetails());

            }
            //void deleteClicked(object sender, EventArgs e)
            //{
            //    var emp = (Employee)BindingContext;
            //    App.Database.DeleteItem(emp.Eid);
            //    this.Navigation.PopAsync();
            //}
            void DetailsClicked(object sender, EventArgs e)
            {
                var empcancel = (Employee)BindingContext;
                this.Navigation.PushAsync(new EmployeeDetails());
            }
            //    void speakClicked(object sender, EventArgs e)
            //    {
            //        var empspek = (Employee)BindingContext;
            //        //DependencyService.Get<ITextSpeak>().Speak(empspek.Address + " " +
empspek.Ename);
            //    }
        }
    }
```

6. um EmployeeDetails hinter dem Code anzuzeigen

```
using DatabaseEmployeeCreation;
using DatabaseEmployeeCreation.SqlLite;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using Xamarin.Forms;

namespace DatabaseEmployeeCreation.SqlLite.Views
{
    public partial class EmployeeDetails : ContentPage
    {
        ListView lv = new ListView();
        IEnumerable<Employee> lst;
        public EmployeeDetails()
        {
            InitializeComponent();
            displayemployee();
        }
```

```
        private void displayemployee()
        {
            Button btn = new Button()
            {

                Text = "Details",
                BackgroundColor = Color.Blue,
            };
            btn.Clicked += Btn_Clicked;
            //IEnumerable<Employee> lst = App.Database.GetItems();
            //IEnumerable<Employee> lst1 = App.Database.GetItemsNotDone();
            //IEnumerable<Employee> lst2 = App.Database.GetItemsNotDone();
            Content = new StackLayout()
            {
                Children = { btn },
            };
        }

        private void Btn_Clicked(object sender, EventArgs e)
        {
            lst = App.Database.GetItems();

            lv.ItemsSource = lst;
            lv.HasUnevenRows = true;
            lv.ItemTemplate = new DataTemplate(typeof(OptionsViewCell));

            Content = new StackLayout()
            {
                Children = { lv },
            };

        }
    }
```

```
    public class OptionsViewCell : ViewCell
    {

        int empid;
        Button btnEdit;
        public OptionsViewCell()
        {
        }
        protected override void OnBindingContextChanged()
        {
            base.OnBindingContextChanged();

            if (this.BindingContext == null)
                return;

            dynamic obj = BindingContext;
            empid = Convert.ToInt32(obj.Eid);
            var lblname = new Label
            {
                BackgroundColor = Color.Lime,
                Text = obj.Ename,
            };

            var lblAddress = new Label
            {
                BackgroundColor = Color.Yellow,
```

```
    Text = obj.Address,
};

var lblphonenumber = new Label
{
    BackgroundColor = Color.Pink,
    Text = obj.phonenumber,
};

var lblemail = new Label
{
    BackgroundColor = Color.Purple,
    Text = obj.email,
};

var lbleid = new Label
{
    BackgroundColor = Color.Silver,
    Text = (empid).ToString(),
};

//var lblname = new Label
//{
//    BackgroundColor = Color.Lime,
//    // HorizontalOptions = LayoutOptions.Start
//};
//lblname.SetBinding(Label.TextProperty, "Ename");

//var lblAddress = new Label
//{
//    BackgroundColor = Color.Yellow,
//    //HorizontalOptions = LayoutOptions.Center,
//};
//lblAddress.SetBinding(Label.TextProperty, "Address");

//var lblphonenumber = new Label
//{
//    BackgroundColor = Color.Pink,
//    //HorizontalOptions = LayoutOptions.CenterAndExpand,
//};
//lblphonenumber.SetBinding(Label.TextProperty, "phonenumber");

//var lblemail = new Label
//{
//    BackgroundColor = Color.Purple,
//    // HorizontalOptions = LayoutOptions.CenterAndExpand
//};
//lblemail.SetBinding(Label.TextProperty, "email");
//var lbleid = new Label
//{
//    BackgroundColor = Color.Silver,
//    // HorizontalOptions = LayoutOptions.CenterAndExpand
//};
//lbleid.SetBinding(Label.TextProperty, "Eid");
Button btnDelete = new Button
{
    BackgroundColor = Color.Gray,

    Text = "Delete",
    //WidthRequest = 15,
    //HeightRequest = 20,
```

```
                    TextColor = Color.Red,
                    HorizontalOptions = LayoutOptions.EndAndExpand,
                };
                btnDelete.Clicked += BtnDelete_Clicked;
                //btnDelete.PropertyChanged += BtnDelete_PropertyChanged;

                btnEdit = new Button
                {
                    BackgroundColor = Color.Gray,
                    Text = "Edit",
                    TextColor = Color.Green,
                };
                // lbleid.SetBinding(Label.TextProperty, "Eid");
                btnEdit.Clicked += BtnEdit_Clicked1; ;
                //btnEdit.Clicked += async (s, e) =>{
                //    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration());
                //};

                View = new StackLayout()
                {
                    Orientation = StackOrientation.Horizontal,
                    BackgroundColor = Color.White,
                    Children = { lbleid, lblname, lblAddress, lblemail, lblphonenumber,
btnDelete, btnEdit },
                };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.Center, WidthRequest = 10,
BackgroundColor = Color.Yellow, Children = { lblAddress } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, WidthRequest = 30, BackgroundColor
= Color.Yellow, Children = { lblemail } };

                //View = new StackLayout()
                //{ HorizontalOptions = LayoutOptions.End, BackgroundColor = Color.Green,
Children = { lblphonenumber } };




                //string Empid =c.eid ;

            }

            private async void BtnEdit_Clicked1(object sender, EventArgs e)
            {
               Employee obj= App.Database.GetItem(empid);
                if (empid > 0)
                {
                    await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(obj));
                }
                else {
                await App.Current.MainPage.Navigation.PushModalAsync(new
EmployeeRegistration(empid));
                }
            }
```

```
            private void BtnDelete_Clicked(object sender, EventArgs e)
            {
                // var eid = Convert.ToInt32(empid);
                // var item = (Xamarin.Forms.Button)sender;
                int eid = empid;
                App.Database.DeleteItem(eid);
            }
            //private void BtnDelete_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
            //{
            // var ename=  e.PropertyName;
            //}
        }

        //private void BtnDelete_Clicked(object sender, EventArgs e)
        //{
        //    var eid = 8;
        //    var item = (Xamarin.Forms.Button)sender;

        //    App.Database.DeleteItem(eid);
        //}
    }
```

7. So implementieren Sie eine Methode in Android und der ios GetConnection () -Methode

```
    using System;
    using Xamarin.Forms;
    using System.IO;
    using DatabaseEmployeeCreation.Droid;
    using DatabaseEmployeeCreation.SqlLite.ViewModel;
    using SQLite;
    using SQLite.Net;

    [assembly: Dependency(typeof(SQLiteEmployee_Andriod))]
    namespace DatabaseEmployeeCreation.Droid
    {
        public class SQLiteEmployee_Andriod : ISQLite
        {
            public SQLiteEmployee_Andriod()
            {
            }

            #region ISQLite implementation
            public SQLiteConnection GetConnection()
            {
                //var sqliteFilename = "EmployeeSQLite.db3";
                //string documentsPath =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal); // Documents
folder
                //var path = Path.Combine(documentsPath, sqliteFilename);

                //// This is where we copy in the prepopulated database
                //Console.WriteLine(path);
                //if (!File.Exists(path))
                //{
                //    var s =
Forms.Context.Resources.OpenRawResource(Resource.Raw.EmployeeSQLite);  // RESOURCE NAME ###

                //    // create a write stream
```

```
                //    FileStream writeStream = new FileStream(path, FileMode.OpenOrCreate,
FileAccess.Write);
                //    // write to the stream
                //    ReadWriteStream(s, writeStream);
                //}

                //var conn = new SQLiteConnection(path);

                //// Return the database connection
                //return conn;
                var filename = "DatabaseEmployeeCreationSQLite.db3";
                var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
                var path = Path.Combine(documentspath, filename);
                var platform = new SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
                var connection = new SQLite.Net.SQLiteConnection(platform, path);
                return connection;
            }

            //public  SQLiteConnection GetConnection()
            //{
            //    var filename = "EmployeeSQLite.db3";
            //    var documentspath =
Environment.GetFolderPath(Environment.SpecialFolder.Personal);
            //    var path = Path.Combine(documentspath, filename);

            //    var platform = new
SQLite.Net.Platform.XamarinAndroid.SQLitePlatformAndroid();
            //    var connection = new SQLite.Net.SQLiteConnection(platform, path);
            //    return connection;
            //}
            #endregion

            /// <summary>
            /// helper method to get the database out of /raw/ and into the user filesystem
            /// </summary>
            void ReadWriteStream(Stream readStream, Stream writeStream)
            {
                int Length = 256;
                Byte[] buffer = new Byte[Length];
                int bytesRead = readStream.Read(buffer, 0, Length);
                // write the required bytes
                while (bytesRead > 0)
                {
                    writeStream.Write(buffer, 0, bytesRead);
                    bytesRead = readStream.Read(buffer, 0, Length);
                }
                readStream.Close();
                writeStream.Close();
            }
        }
    }
```

Ich hoffe, dass dieses Beispiel sehr einfach ist

Mit lokalen Datenbanken arbeiten online lesen: https://riptutorial.com/de/xamarin-forms/topic/5997/mit-lokalen-datenbanken-arbeiten

Kapitel 25: Mitteilungen

Bemerkungen

Es gibt keine einheitliche Möglichkeit, Push-Benachrichtigungen in Xamarin Forms zu verarbeiten, da die Implementierung stark von plattformspezifischen Funktionen und Ereignissen abhängt. Daher ist immer plattformspezifischer Code erforderlich.

Mit dem `DependencyService` Sie jedoch so viel Code wie möglich freigeben. Es gibt auch ein von rdelrosario dafür entworfenes Plugin, das auf seinem GitHub zu finden ist .

Code und Screenshots stammen aus einer Blog-Serie von Gerald Versluis, in der der Ablauf näher erläutert wird.

Examples

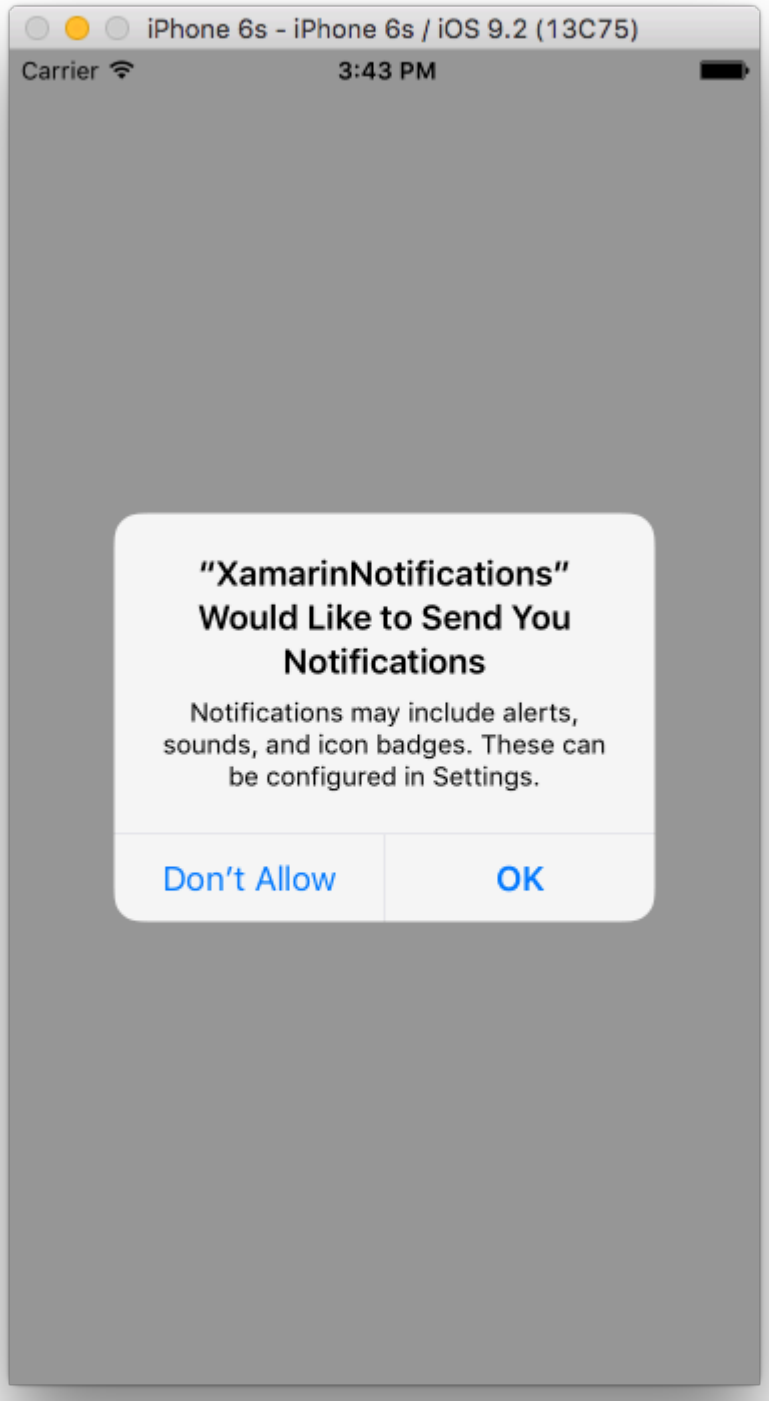
Push-Benachrichtigungen für iOS mit Azure

Um die Registrierung für Push-Benachrichtigungen zu starten, müssen Sie den folgenden Code ausführen.

```
// registers for push
var settings = UIUserNotificationSettings.GetSettingsForTypes(
    UIUserNotificationType.Alert
    | UIUserNotificationType.Badge
    | UIUserNotificationType.Sound,
    new NSSet());

UIApplication.SharedApplication.RegisterUserNotificationSettings(settings);
UIApplication.SharedApplication.RegisterForRemoteNotifications();
```

Dieser Code kann entweder direkt ausgeführt werden, wenn die App im `FinishedLaunching` in der Datei `AppDelegate.cs` wird. Sie können dies auch tun, wenn ein Benutzer entscheidet, Push-Benachrichtigungen zu aktivieren.

Wenn Sie diesen Code ausführen, wird eine Warnmeldung ausgegeben, die den Benutzer auffordert, zu akzeptieren, dass die App Benachrichtigungen senden kann. Implementieren Sie auch ein Szenario, in dem der Benutzer dies ablehnt!

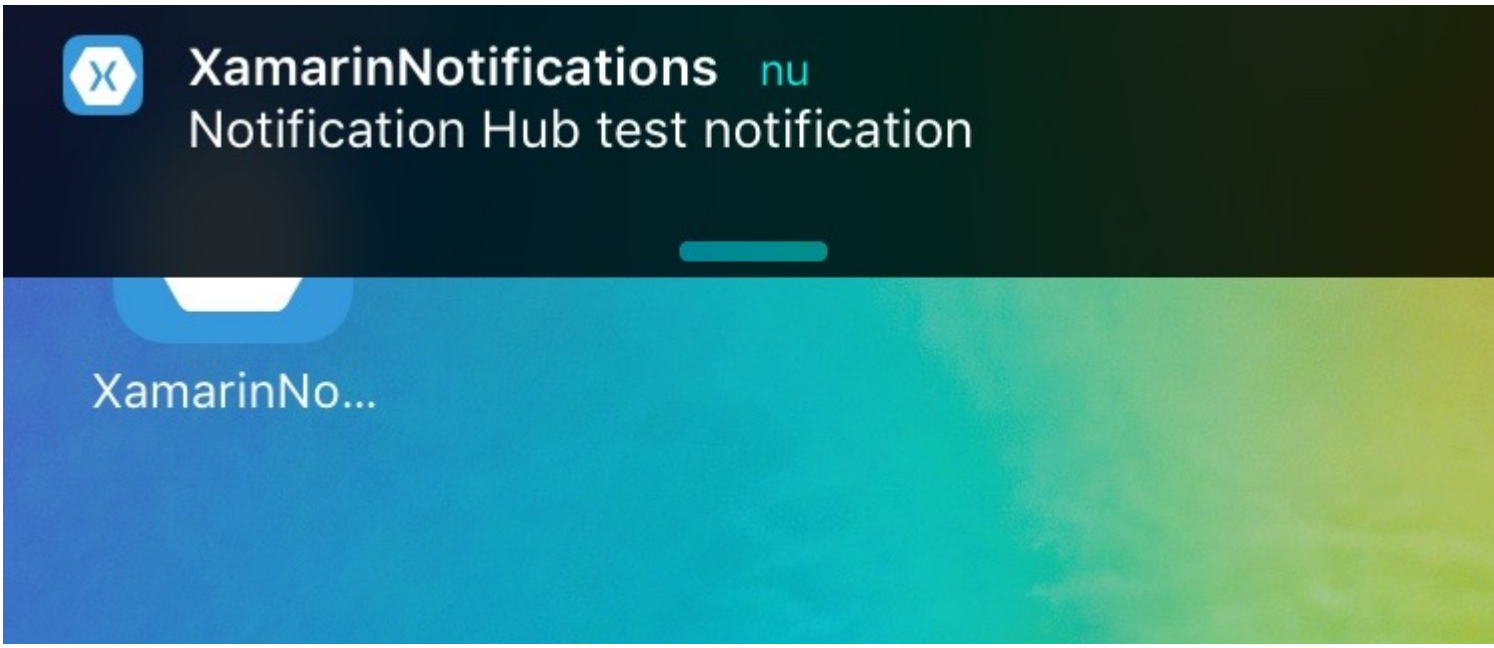


Dies sind die Ereignisse, die zur Implementierung von Push-Benachrichtigungen unter iOS implementiert werden müssen. Sie finden sie in der `AppDelegate.cs` Datei.

```
// We've successfully registered with the Apple notification service, or in our case Azure
public override void RegisteredForRemoteNotifications(UIApplication application, NSData
deviceToken)
{
   // Modify device token for compatibility Azure
    var token = deviceToken.Description;
```

```
    token = token.Trim('<', '>').Replace(" ", "");

    // You need the Settings plugin for this!
    Settings.DeviceToken = token;

    var hub = new SBNotificationHub("Endpoint=sb://xamarinnotifications-
ns.servicebus.windows.net/;SharedAccessKeyName=DefaultListenSharedAccessSignature;SharedAccessKey=<your
own key>", "xamarinnotifications");

    NSSet tags = null; // create tags if you want, not covered for now
    hub.RegisterNativeAsync(deviceToken, tags, (errorCallback) =>
    {
        if (errorCallback != null)
        {
            var alert = new UIAlertView("ERROR!", errorCallback.ToString(), null, "OK", null);
            alert.Show();
        }
    });
}

// We've received a notification, yay!
public override void ReceivedRemoteNotification(UIApplication application, NSDictionary
userInfo)
{
    NSObject inAppMessage;

    var success = userInfo.TryGetValue(new NSString("inAppMessage"), out inAppMessage);

    if (success)
    {
        var alert = new UIAlertView("Notification!", inAppMessage.ToString(), null, "OK",
null);
        alert.Show();
    }
}

// Something went wrong while registering!
public override void FailedToRegisterForRemoteNotifications(UIApplication application, NSError
error)
{
   var alert = new UIAlertView("Computer says no", "Notification registration failed! Try
again!", null, "OK", null);

   alert.Show();
}
```

Wenn eine Benachrichtigung eingeht, sieht es so aus.



Push-Benachrichtigungen für Android mit Azure

Die Implementierung auf Android ist etwas aufwendiger und erfordert die Implementierung eines bestimmten `Service` .

Lassen Sie uns zunächst prüfen, ob unser Gerät Push-Benachrichtigungen empfangen kann. Wenn ja, registrieren Sie es bei Google. Dies kann mit diesem Code in unserer Datei `MainActivity.cs` .

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    global::Xamarin.Forms.Forms.Init(this, bundle);

    // Check to ensure everything's setup right for push
    GcmClient.CheckDevice(this);
    GcmClient.CheckManifest(this);
    GcmClient.Register(this, NotificationsBroadcastReceiver.SenderIDs);

    LoadApplication(new App());
}
```

Die SenderIDs befinden sich im Code darunter und sind die Projektnummer, die Sie vom Google Entwickler-Dashboard erhalten, um Push-Nachrichten senden zu können.

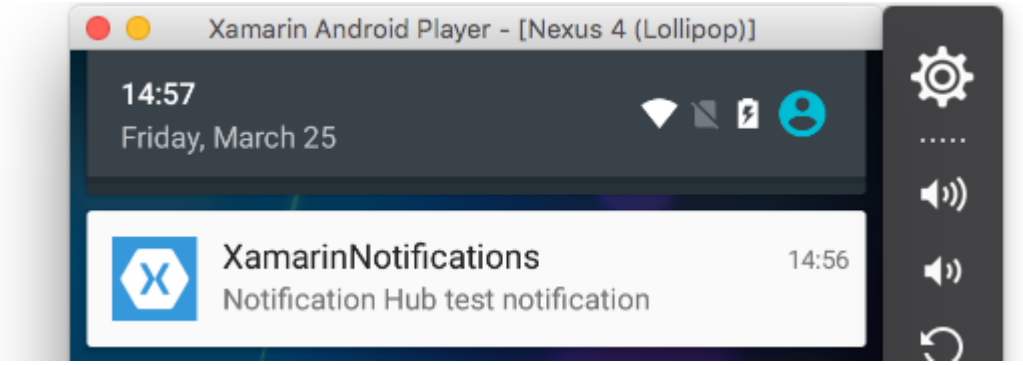
```
using Android.App;
using Android.Content;
using Gcm.Client;
using Java.Lang;
using System;
using WindowsAzure.Messaging;
using XamarinNotifications.Helpers;

// These attributes are to register the right permissions for our app concerning push messages
[assembly: Permission(Name = "com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
```

```
[assembly: UsesPermission(Name =
"com.versluisit.xamarinnotifications.permission.C2D_MESSAGE")]
[assembly: UsesPermission(Name = "com.google.android.c2dm.permission.RECEIVE")]

//GET_ACCOUNTS is only needed for android versions 4.0.3 and below
[assembly: UsesPermission(Name = "android.permission.GET_ACCOUNTS")]
[assembly: UsesPermission(Name = "android.permission.INTERNET")]
[assembly: UsesPermission(Name = "android.permission.WAKE_LOCK")]

namespace XamarinNotifications.Droid.PlatformSpecifics
{
    // These attributes belong to the BroadcastReceiver, they register for the right intents
    [BroadcastReceiver(Permission = Constants.PERMISSION_GCM_INTENTS)]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_MESSAGE },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_REGISTRATION_CALLBACK },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]
    [IntentFilter(new[] { Constants.INTENT_FROM_GCM_LIBRARY_RETRY },
    Categories = new[] { "com.versluisit.xamarinnotifications" })]

    // This is the bradcast reciever
    public class NotificationsBroadcastReceiver : GcmBroadcastReceiverBase<PushHandlerService>
    {
        // TODO add your project number here
        public static string[] SenderIDs = { "96688------" };
    }

    [Service] // Don't forget this one! This tells Xamarin that this class is a Android
Service
    public class PushHandlerService : GcmServiceBase
    {
        // TODO add your own access key
        private string _connectionString =
ConnectionString.CreateUsingSharedAccessKeyWithListenAccess(
            new Java.Net.URI("sb://xamarinnotifications-ns.servicebus.windows.net/"), "<your
key here>");

        // TODO add your own hub name
        private string _hubName = "xamarinnotifications";

        public static string RegistrationID { get; private set; }

        public PushHandlerService() : base(NotificationsBroadcastReceiver.SenderIDs)
        {
        }

        // This is the entry point for when a notification is received
        protected override void OnMessage(Context context, Intent intent)
        {
            var title = "XamarinNotifications";

            if (intent.Extras.ContainsKey("title"))
                title = intent.Extras.GetString("title");

            var messageText = intent.Extras.GetString("message");

            if (!string.IsNullOrEmpty(messageText))
                CreateNotification(title, messageText);
        }

        // The method we use to compose our notification
```

```
private void CreateNotification(string title, string desc)
{
    // First we make sure our app will start when the notification is pressed
    const int pendingIntentId = 0;
    const int notificationId = 0;

    var startupIntent = new Intent(this, typeof(MainActivity));
    var stackBuilder = TaskStackBuilder.Create(this);

    stackBuilder.AddParentStack(Class.FromType(typeof(MainActivity)));
    stackBuilder.AddNextIntent(startupIntent);

    var pendingIntent =
        stackBuilder.GetPendingIntent(pendingIntentId, PendingIntentFlags.OneShot);

    // Here we start building our actual notification, this has some more
    // interesting customization options!
    var builder = new Notification.Builder(this)
        .SetContentIntent(pendingIntent)
        .SetContentTitle(title)
        .SetContentText(desc)
        .SetSmallIcon(Resource.Drawable.icon);

    // Build the notification
    var notification = builder.Build();
    notification.Flags = NotificationFlags.AutoCancel;

    // Get the notification manager
    var notificationManager =
        GetSystemService(NotificationService) as NotificationManager;

    // Publish the notification to the notification manager
    notificationManager.Notify(notificationId, notification);
}

// Whenever an error occurs in regard to push registering, this fires
protected override void OnError(Context context, string errorId)
{
    Console.Out.WriteLine(errorId);
}

// This handles the successful registration of our device to Google
// We need to register with Azure here ourselves
protected override void OnRegistered(Context context, string registrationId)
{
    var hub = new NotificationHub(_hubName, _connectionString, context);

    Settings.DeviceToken = registrationId;

    // TODO set some tags here if you want and supply them to the Register method
    var tags = new string[] { };

    hub.Register(registrationId, tags);
}

// This handles when our device unregisters at Google
// We need to unregister with Azure
protected override void OnUnRegistered(Context context, string registrationId)
{
    var hub = new NotificationHub(_hubName, _connectionString, context);
```

```
            hub.UnregisterAll(registrationId);
        }
    }
}
```

Eine Beispielbenachrichtigung auf Android sieht so aus.

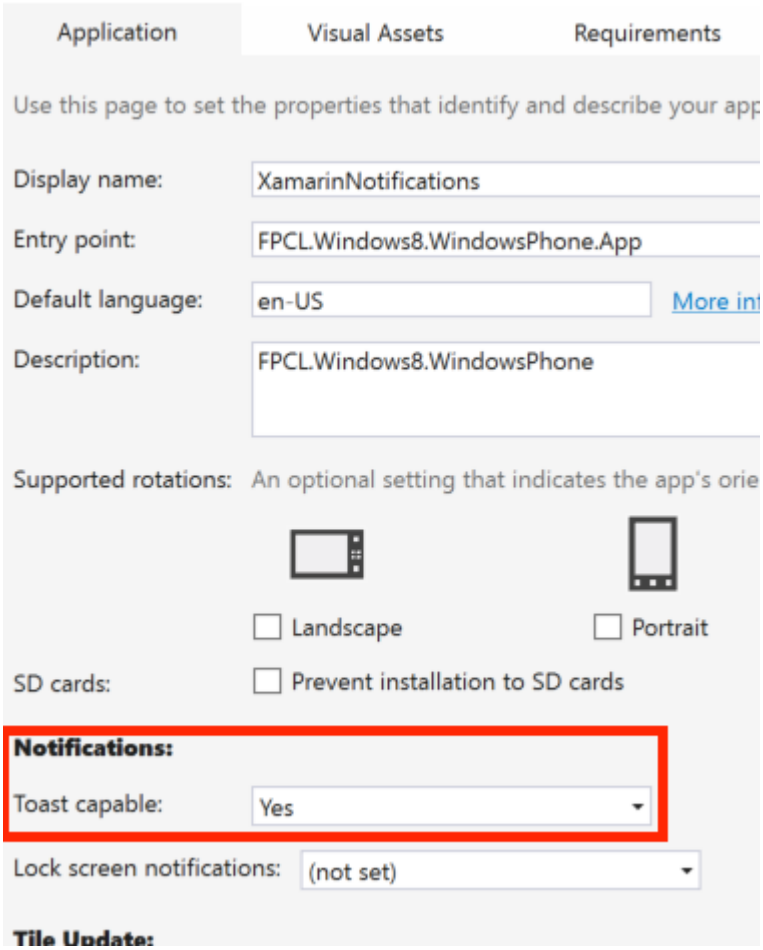


Push-Benachrichtigungen für Windows Phone mit Azure

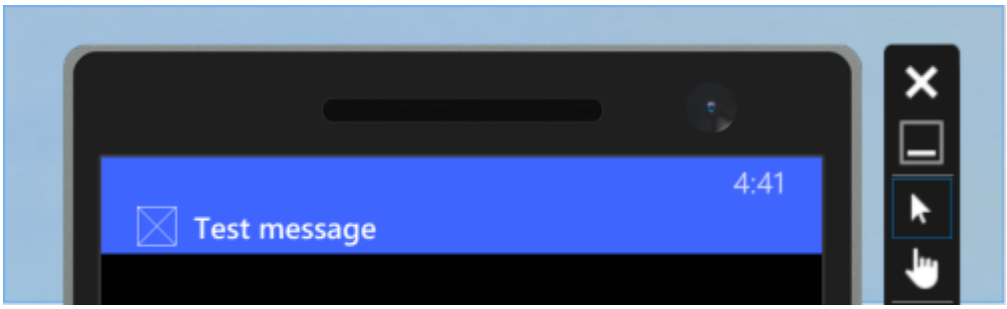
Unter Windows Phone muss etwas wie der Code darunter implementiert werden, um mit Push-Benachrichtigungen arbeiten zu können. Dies kann in der `App.xaml.cs` Datei gefunden werden.

```
protected async override void OnLaunched(LaunchActivatedEventArgs e)
{
    var channel = await
PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();

    // TODO add connection string here
    var hub = new NotificationHub("XamarinNotifications", "<connection string with listen
access>");
    var result = await hub.RegisterNativeAsync(channel.Uri);

    // Displays the registration ID so you know it was successful
    if (result.RegistrationId != null)
    {
        Settings.DeviceToken = result.RegistrationId;
    }

    // The rest of the default code is here
}
```

Vergessen Sie auch nicht, die Funktionen in der `Package.appxmanifest` Datei zu `Package.appxmanifest` .



Eine Beispiel-Push-Benachrichtigung kann wie folgt aussehen:



Mitteilungen online lesen: https://riptutorial.com/de/xamarin-forms/topic/5042/mitteilungen

Kapitel 26: Mitteilungen

Bemerkungen

AWS Simple Benachrichtigungsdienst Lingo:

Endpunkt - Der Endpunkt kann ein Telefon, eine E-Mail-Adresse oder was auch immer sein. Dies ist, was AWS SNS mit einer Benachrichtigung zurückschlagen kann

Thema - Im Wesentlichen eine Gruppe, die alle Ihre Endpunkte enthält

Abonnieren - Sie melden sich bei Ihrem Telefon / Client an, um Benachrichtigungen zu erhalten

Generisches Pushbenachrichtigungs-Lingo:

APNS - Apple Push Benachrichtigungsdienst. Apple kann als einziger Push-Benachrichtigungen senden. Deshalb versorgen wir unsere App mit dem richtigen Zertifikat. Wir stellen AWS SNS das Zertifikat zur Verfügung, das uns von Apple erteilt wird, um SNS zu autorisieren, in unserem Namen eine Benachrichtigung an APNS zu senden.

GCM - Google Cloud Messaging ist APNS sehr ähnlich. Nur Google kann Push-Benachrichtigungen direkt versenden. Also registrieren wir zuerst unsere App in GCM und übergeben unser Token an AWS SNS. SNS kümmert sich um alles, was mit GCM zu tun hat und die Daten überträgt.

Examples

iOS-Beispiel

1. Sie benötigen ein Entwicklungsgerät
2. Wechseln Sie zu Ihrem Apple Developer Account und erstellen Sie ein Bereitstellungsprofil mit aktivierten Push-Benachrichtigungen
3. Sie werden irgendeine Art von Art und Weise müssen Sie Ihr Telefon (AWS, Azure..etc) benachrichtigen **Wir AWS nutzen werden hier**

```
public override bool FinishedLaunching(UIApplication app, NSDictionary options)
{
    global::Xamarin.Forms.Forms.Init();

   //after typical Xamarin.Forms Init Stuff

   //variable to set-up the style of notifications you want, iOS supports 3 types

   var pushSettings = UIUserNotificationSettings.GetSettingsForTypes(
              UIUserNotificationType.Alert |
              UIUserNotificationType.Badge |
              UIUserNotificationType.Sound,
```

```
            null );


        //both of these methods are in iOS, we have to override them and set them up
        //to allow push notifications

        app.RegisterUserNotificationSettings(pushSettings);  //pass the supported push
notifications settings to register app in settings page


}

public override async void RegisteredForRemoteNotifications(UIApplication application, NSData
token)
    {
        AmazonSimpleNotificationServiceClient snsClient = new
AmazonSimpleNotificationServiceClient("your AWS credentials here");

        // This contains the registered push notification token stored on the phone.
        var deviceToken = token.Description.Replace("<", "").Replace(">", "").Replace(" ",
"");

        if (!string.IsNullOrEmpty(deviceToken))
        {
            //register with SNS to create an endpoint ARN, this means AWS can message your
phone
            var response = await snsClient.CreatePlatformEndpointAsync(
            new CreatePlatformEndpointRequest
            {
                Token = deviceToken,
                PlatformApplicationArn = "yourARNwouldgohere" /* insert your platform
application ARN here */
            });

            var endpoint = response.EndpointArn;

            //AWS lets you create topics, so use subscribe your app to a topic, so you can
easily send out one push notification to all of your users
            var subscribeResponse = await snsClient.SubscribeAsync(new SubscribeRequest
            {
                TopicArn = "YourTopicARN here",
                Endpoint = endpoint,
                Protocol = "application"

            });

        }

    }
```

Mitteilungen online lesen: https://riptutorial.com/de/xamarin-forms/topic/5998/mitteilungen

Kapitel 27: Navigation in Xamarin.Forms

Examples

NavigationSeitenfluss

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public class App : Application
    {
        public App()
        {
            MainPage = new NavigationPage(new FirstPage());
        }
    }

    public class FirstPage : ContentPage
    {
        Label FirstPageLabel { get; set; } = new Label();

        Button FirstPageButton { get; set; } = new Button();

        public FirstPage()
        {
            Title = "First page";

            FirstPageLabel.Text = "This is the first page";
            FirstPageButton.Text = "Navigate to the second page";
            FirstPageButton.Clicked += OnFirstPageButtonClicked;

            var content = new StackLayout();
            content.Children.Add(FirstPageLabel);
            content.Children.Add(FirstPageButton);

            Content = content;
        }

        async void OnFirstPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }
    }

    public class SecondPage : ContentPage
    {
        Label SecondPageLabel { get; set; } = new Label();

        public SecondPage()
        {
            Title = "Second page";

            SecondPageLabel.Text = "This is the second page";

            Content = SecondPageLabel;
```

```
        }
    }
}
```

Navigationsseitenfluss mit XAML

App.xaml.cs-Datei (App.xaml-Datei ist Standard, wird also übersprungen)

```
using Xamrin.Forms

namespace NavigationApp
{
    public partial class App : Application
    {
        public static INavigation GlobalNavigation { get; private set; }

        public App()
        {
            InitializeComponent();
            var rootPage = new NavigationPage(new FirstPage());

            GlobalNavigation = rootPage.Navigation;

            MainPage = rootPage;
        }
    }
}
```

Datei "FirstPage.xaml"

```
<?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="NavigationApp.FirstPage"
    Title="First page">
    <ContentPage.Content>
        <StackLayout>
            <Label
                Text="This is the first page" />
            <Button
                Text="Click to navigate to a new page"
                Clicked="GoToSecondPageButtonClicked"/>
            <Button
                Text="Click to open the new page as modal"
                Clicked="OpenGlobalModalPageButtonClicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

In einigen Fällen müssen Sie die neue Seite nicht in der aktuellen, sondern in der globalen Navigation öffnen. Wenn Ihre aktuelle Seite beispielsweise ein unteres Menü enthält, wird es sichtbar, wenn Sie die neue Seite in der aktuellen Navigation drücken. Wenn die Seite über den gesamten sichtbaren Inhalt geöffnet werden muss, um das untere Menü und den Inhalt der aktuellen Seite auszublenden, müssen Sie die neue Seite als modal in die globale Navigation

einfügen. Siehe `App.GlobalNavigation` Eigenschaft und das folgende Beispiel.

Datei FirstPage.xaml.cs

```
using System;
using Xamarin.Forms;

namespace NavigationApp
{
    public partial class FirstPage : ContentPage
    {
        public FirstPage()
        {
            InitializeComponent();
        }

        async void GoToSecondPageButtonClicked(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new SecondPage(), true);
        }

        async void OpenGlobalModalPageButtonClicked(object sender, EventArgs e)
        {
            await App.GlobalNavigation.PushModalAsync(new SecondPage(), true);
        }
    }
}
```

SecondPage.xaml-Datei (xaml.cs-Datei ist Standard, wird also übersprungen)

```
<?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="NavigationApp.SecondPage"
    Title="Second page">
    <ContentPage.Content>
        <Label
            Text="This is the second page" />
    </ContentPage.Content>
</ContentPage>
```

Hierarchische Navigation mit XAML

Standardmäßig funktioniert das Navigationsmuster wie ein Stapel von Seiten und ruft die neuesten Seiten über die vorherigen Seiten auf. Sie müssen dazu das NavigationPage- Objekt verwenden.

Neue Seiten schieben

```
...
public class App : Application
{
    public App()
    {
```

```
        MainPage = new NavigationPage(new Page1());
    }
}
...
```

Page1.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 1" />
        <Button Text="Go to page 2" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page1.xaml.cs

```
...
public partial class Page1 : ContentPage
{
    public Page1()
    {
        InitializeComponent();
    }

    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page2());
    }
}
...
```

Page2.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 2" />
        <Button Text="Go to Page 3" Clicked="GoToNextPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page2.xaml.cs

```
...
public partial class Page2 : ContentPage
{
    public Page2()
    {
        InitializeComponent();
    }
```

```
    protected async void GoToNextPage(object sender, EventArgs e)
    {
        await Navigation.PushAsync(new Page3());
    }
}
...
```

Seiten knallen

Normalerweise verwendet der Benutzer die Zurück-Taste, um Seiten zurückzugeben, aber manchmal müssen Sie dies programmgesteuert steuern. Daher müssen Sie die Methode **NavigationPage.PopAsync ()** aufrufen, um zur vorherigen Seite zurückzukehren, oder **NavigationPage.PopToRootAsync ().** so wie ...

Page3.xaml

```
...
<ContentPage.Content>
    <StackLayout>
        <Label Text="Page 3" />
        <Button Text="Go to previous page" Clicked="GoToPreviousPage" />
        <Button Text="Go to beginning" Clicked="GoToStartPage" />
    </StackLayout>
</ContentPage.Content>
...
```

Page3.xaml.cs

```
...
public partial class Page3 : ContentPage
{
    public Page3()
    {
        InitializeComponent();
    }

    protected async void GoToPreviousPage(object sender, EventArgs e)
    {
        await Navigation.PopAsync();
    }

    protected async void GoToStartPage(object sender, EventArgs e)
    {
        await Navigation.PopToRootAsync();
    }
}
...
```

Modale Navigation mit XAML

Modalseiten können auf drei Arten erstellt werden:

- Aus dem **NavigationPage-** Objekt für Vollbildseiten

- Für Warnungen und Benachrichtigungen
- Für Action Sheets, die Popup-Menüs sind

Vollbildmodale

```
...
// to open
await Navigation.PushModalAsync(new ModalPage());
// to close
await Navigation.PopModalAsync();
...
```

Alarme / Bestätigungen und Benachrichtigungen

```
...
// alert
await DisplayAlert("Alert title", "Alert text", "Ok button text");
// confirmation
var booleanAnswer = await DisplayAlert("Confirm?", "Confirmation text", "Yes", "No");
...
```

Aktionsblätter

```
...
var selectedOption = await DisplayActionSheet("Options", "Cancel", "Destroy", "Option 1",
"Option 2", "Option 3");
...
```

Master-Detailseite

```
public class App : Application
{
    internal static NavigationPage NavPage;

    public App ()
    {
        // The root page of your application
        MainPage = new RootPage();
    }
}
public class RootPage : MasterDetailPage
{
    public RootPage()
    {
        var menuPage = new MenuPage();
        menuPage.Menu.ItemSelected += (sender, e) => NavigateTo(e.SelectedItem as MenuItem);
        Master = menuPage;
        App.NavPage = new NavigationPage(new HomePage());
        Detail = App.NavPage;
    }
    protected override async void OnAppearing()
    {
```

```
        base.OnAppearing();
    }
    void NavigateTo(MenuItem menuItem)
    {
        Page displayPage = (Page)Activator.CreateInstance(menuItem.TargetType);
        Detail = new NavigationPage(displayPage);
        IsPresented = false;
    }
}
```

Master Detail Navigation

Der folgende Code zeigt, wie eine asynchrone Navigation durchgeführt wird, wenn sich die App in einem MasterDetailPage-Kontext befindet.

```
public async Task NavigateMasterDetail(Page page)
{
        if (page == null)
            {
                return;
            }

        var masterDetail = App.Current.MainPage as MasterDetailPage;

        if (masterDetail == null || masterDetail.Detail == null)
            return;

        var navigationPage = masterDetail.Detail as NavigationPage;
        if (navigationPage == null)
        {
            masterDetail.Detail = new NavigationPage(page);
            masterDetail.IsPresented = false;
            return;
        }

        await navigationPage.Navigation.PushAsync(page);


navigationPage.Navigation.RemovePage(navigationPage.Navigation.NavigationStack[navigationPage.Navigatio
- 2]);
        masterDetail.IsPresented = false;
    }
```

Navigation in Xamarin.Forms online lesen: https://riptutorial.com/de/xamarin-forms/topic/1571/navigation-in-xamarin-forms

Kapitel 28: Navigation in Xamarin.Forms

Bemerkungen

Die Navigation auf Xamarin.Forms basiert auf zwei Hauptnavigationsmustern: hierarchisch und modal.

Das hierarchische Muster ermöglicht es dem Benutzer, sich in einem Stapel von Seiten nach unten zu bewegen und durch Drücken der Taste "Zurück" / "Hoch" zurückzukehren.

Das modale Muster ist eine Unterbrechungsseite, die vom Benutzer eine bestimmte Aktion erfordert, die aber normalerweise durch Drücken der Abbruchtaste abgebrochen werden kann. Beispiele sind Benachrichtigungen, Alarme, Dialogfelder und Register / Editionsseiten.

Examples

Verwendung von INavigation aus dem Ansichtsmodell

Der erste Schritt ist das Erstellen einer Navigationsschnittstelle, die wir für das Ansichtsmodell verwenden werden:

```
public interface IViewNavigationService
{
    void Initialize(INavigation navigation, SuperMapper navigationMapper);
    Task NavigateToAsync(object navigationSource, object parameter = null);
    Task GoBackAsync();
}
```

Bei der `Initialize` verwende ich meinen benutzerdefinierten Mapper, bei dem ich die Seitentypen mit den zugehörigen Schlüsseln sammle.

```
public class SuperMapper
{
    private readonly ConcurrentDictionary<Type, object> _typeToAssociateDictionary = new
ConcurrentDictionary<Type, object>();

    private readonly ConcurrentDictionary<object, Type> _associateToType = new
ConcurrentDictionary<object, Type>();

    public void AddMapping(Type type, object associatedSource)
    {
        _typeToAssociateDictionary.TryAdd(type, associatedSource);
        _associateToType.TryAdd(associatedSource, type);
    }

    public Type GetTypeSource(object associatedSource)
    {
        Type typeSource;
        _associateToType.TryGetValue(associatedSource, out typeSource);
```

```
        return typeSource;
    }

    public object GetAssociatedSource(Type typeSource)
    {
        object associatedSource;
        _typeToAssociateDictionary.TryGetValue(typeSource, out associatedSource);

        return associatedSource;
    }
}
```

Aufzählung mit Seiten:

```
public enum NavigationPageSource
{
    Page1,
    Page2
}
```

`App.cs` Datei:

```
public class App : Application
{
    public App()
    {
        var startPage = new Page1();
        InitializeNavigation(startPage);
        MainPage = new NavigationPage(startPage);
    }

    #region Sample of navigation initialization
    private void InitializeNavigation(Page startPage)
    {
        var mapper = new SuperMapper();
        mapper.AddMapping(typeof(Page1), NavigationPageSource.Page1);
        mapper.AddMapping(typeof(Page2), NavigationPageSource.Page2);

        var navigationService = DependencyService.Get<IViewNavigationService>();
        navigationService.Initialize(startPage.Navigation, mapper);
    }
    #endregion
}
```

In Mapper verknüpfte ich den Typ einer Seite mit dem Aufzählungswert.

`IViewNavigationService` Implementierung:

```
[assembly: Dependency(typeof(ViewNavigationService))]
namespace SuperForms.Core.ViewNavigation
{
    public class ViewNavigationService : IViewNavigationService
    {
        private INavigation _navigation;
        private SuperMapper _navigationMapper;

        public void Initialize(INavigation navigation, SuperMapper navigationMapper)
```

```
    {
        _navigation = navigation;
        _navigationMapper = navigationMapper;
    }

    public async Task NavigateToAsync(object navigationSource, object parameter = null)
    {
        CheckIsInitialized();

        var type = _navigationMapper.GetTypeSource(navigationSource);

        if (type == null)
        {
            throw new InvalidOperationException(
                "Can't find associated type for " + navigationSource.ToString());
        }

        ConstructorInfo constructor;
        object[] parameters;

        if (parameter == null)
        {
            constructor = type.GetTypeInfo()
                              .DeclaredConstructors
                              .FirstOrDefault(c => !c.GetParameters().Any());

            parameters = new object[] { };
        }
        else
        {
            constructor = type.GetTypeInfo()
                              .DeclaredConstructors
                              .FirstOrDefault(c =>
                                {
                                    var p = c.GetParameters();
                                    return p.Count() == 1 &&
                                        p[0].ParameterType == parameter.GetType();
                                });

            parameters = new[] { parameter };
        }

        if (constructor == null)
        {
            throw new InvalidOperationException(
                "No suitable constructor found for page " + navigationSource.ToString());
        }

        var page = constructor.Invoke(parameters) as Page;

        await _navigation.PushAsync(page);
    }

    public async Task GoBackAsync()
    {
        CheckIsInitialized();

        await _navigation.PopAsync();
    }

    private void CheckIsInitialized()
```

```
        {
            if (_navigation == null || _navigationMapper == null)
                throw new NullReferenceException("Call Initialize method first.");
        }
    }
}
```

Ich bekomme den Seitentyp, auf dem der Benutzer navigieren möchte, und erzeuge dessen Instanz mithilfe von Reflektion.

Und dann könnte ich den Navigationsdienst für das Ansichtsmodell verwenden:

```
var navigationService = DependencyService.Get<IViewNavigationService>();
await navigationService.NavigateToAsync(NavigationPageSource.Page2, "hello from Page1");
```

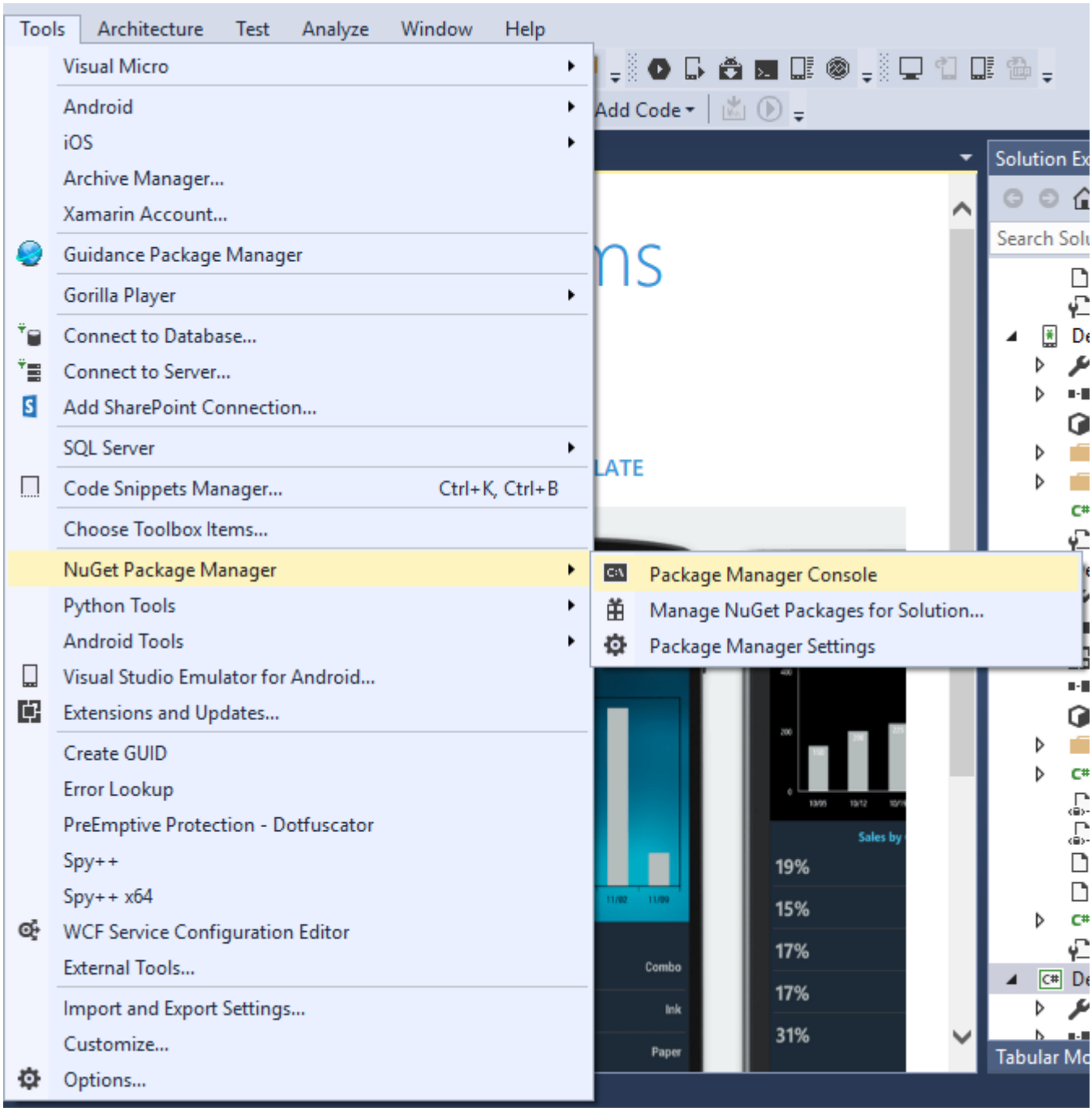
Navigation in Xamarin.Forms online lesen: https://riptutorial.com/de/xamarin-forms/topic/2507/navigation-in-xamarin-forms

Kapitel 29: OAuth2

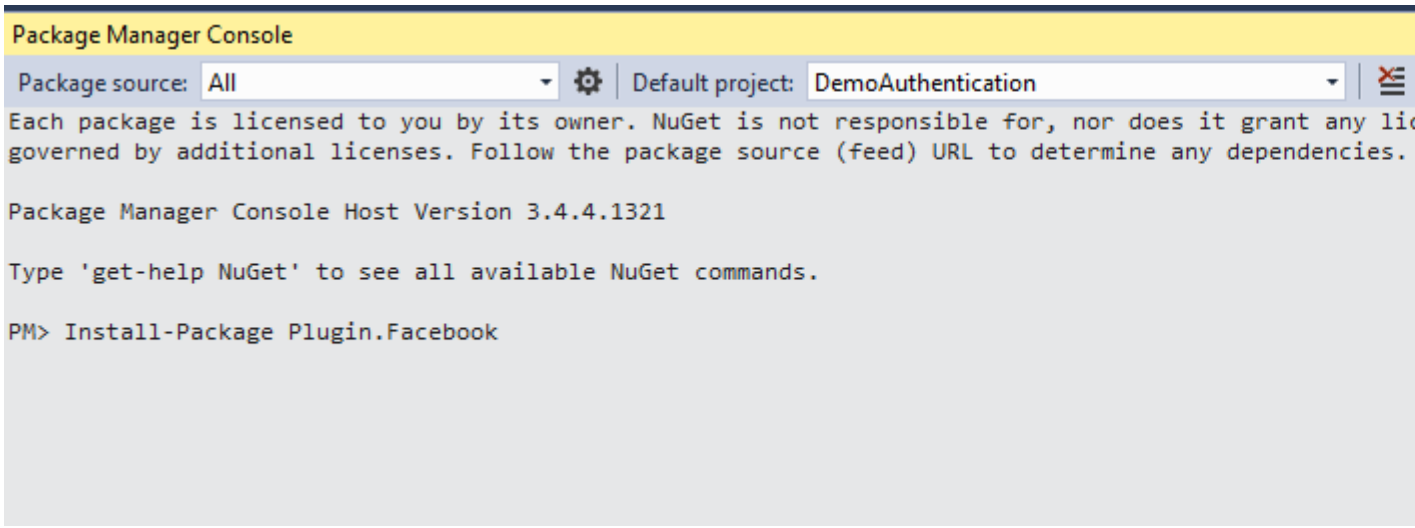
Examples

Authentifizierung mit dem Plugin

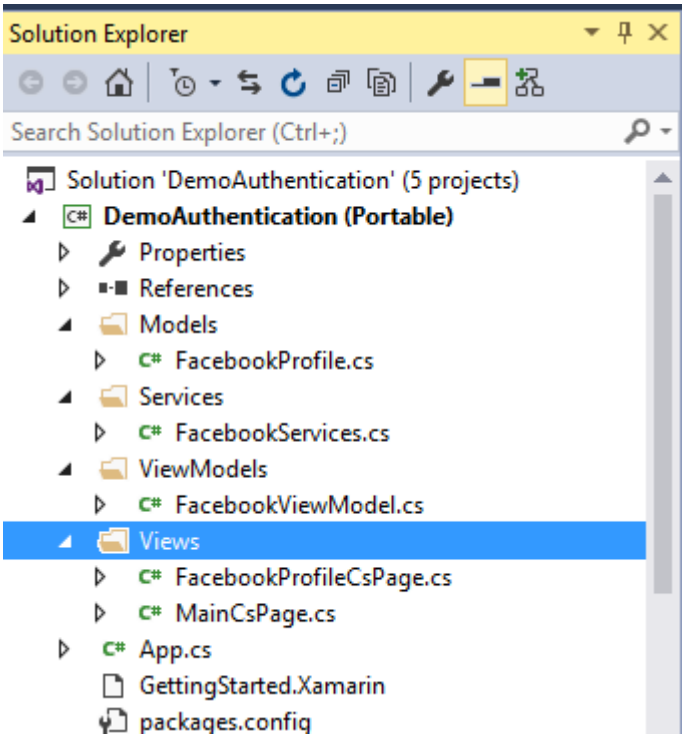
1. Gehen Sie zuerst zu **Tools** > **NuGet Package Manager** > **Package Manager Console** .



2. Geben Sie diesen Befehl " **Install-Package Plugin.Facebook** " in der Package **Manager-**Konsole ein.



3. Nun wird die gesamte Datei automatisch erstellt.



Video : Mit Facebook in Xamarin-Formularen anmelden

Andere Authentifizierung mit dem Plugin. Platzieren Sie den Befehl wie in Schritt 2 gezeigt in der Package Manager Console.

1. **Youtube** : Install-Package Plugin.Youtube
2. **Twitter** : Install-Package Plugin.Twitter
3. **Foursquare** : Install-Package Plugin.Foursquare
4. **Google** : Install-Package Plugin.Google
5. **Instagram** : Install-Package Plugin.Instagram
6. **Eventbrite** : Install-Package Plugin.Eventbrite

OAuth2 online lesen: https://riptutorial.com/de/xamarin-forms/topic/8828/oauth2

Kapitel 30: Plattformspezifische visuelle Anpassungen

Examples

Sprachanpassungen

Idiomspezifische Anpassungen können über den C # -Code vorgenommen werden, z. B. zum Ändern der Layout-Ausrichtung, unabhängig davon, ob die Ansicht angezeigt wird, ein Telefon oder ein Tablet.

```
if (Device.Idiom == TargetIdiom.Phone)
{
    this.panel.Orientation = StackOrientation.Vertical;
}
else
{
    this.panel.Orientation = StackOrientation.Horizontal;
}
```

Diese Funktionalitäten sind auch direkt aus XAML-Code verfügbar:

```
<StackLayout x:Name="panel">
  <StackLayout.Orientation>
    <OnIdiom x:TypeArguments="StackOrientation">
      <OnIdiom.Phone>Vertical</OnIdiom.Phone>
      <OnIdiom.Tablet>Horizontal</OnIdiom.Tablet>
    </OnIdiom>
  </StackLayout.Orientation>
</StackLayout>
```

Plattformanpassungen

Anpassungen können für bestimmte Plattformen vom C # -Code aus vorgenommen werden, z. B. zum Ändern der Auffüllung für alle Zielplattformen.

```
if (Device.OS == TargetPlatform.iOS)
{
    panel.Padding = new Thickness (10);
}
else
{
    panel.Padding = new Thickness (20);
}
```

Für verkürzte C # -Deklarationen steht auch eine Hilfsmethode zur Verfügung:

```
panel.Padding = new Thickness (Device.OnPlatform(10,20,0));
```

Diese Funktionalitäten sind auch direkt aus XAML-Code verfügbar:

```
<StackLayout x:Name="panel">
  <StackLayout.Padding>
    <OnPlatform x:TypeArguments="Thickness"
      iOS="10"
      Android="20" />
  </StackLayout.Padding>
</StackLayout>
```

Stile verwenden

Wenn Sie mit XAML arbeiten, können Sie mit einem zentralisierten `Style` eine Reihe von Stilansichten von einem Ort aus aktualisieren. Alle Sprach- und Plattformeinstellungen können auch in Ihre Styles integriert werden.

```
<Style TargetType="StackLayout">
  <Setter Property="Padding">
    <Setter.Value>
      <OnPlatform x:TypeArguments="Thickness"
                  iOS="10"
                  Android="20"/>
    </Setter.Value>
  </Setter>
</Style>
```

Benutzerdefinierte Ansichten verwenden

Sie können benutzerdefinierte Ansichten erstellen, die dank dieser Anpassungswerkzeuge in Ihre Seite integriert werden können.

Wählen Sie `File > New > File... > Forms > Forms ContentView (Xaml)` und erstellen Sie eine Ansicht für jedes bestimmte Layout: `TabletHome.xaml` und `PhoneHome.xaml` .

`File > New > File... > Forms > Forms ContentPage` dann `File > New > File... > Forms > Forms ContentPage` und erstellen Sie eine `HomePage.cs` , die `File > New > File... > Forms > Forms ContentPage` enthält:

```
using Xamarin.Forms;

public class HomePage : ContentPage
{
    public HomePage()
    {
        if (Device.Idiom == TargetIdiom.Phone)
        {
            Content = new PhoneHome();
        }
        else
        {
            Content = new TabletHome();
        }

    }
```

```
}
```

Sie haben jetzt eine `HomePage` , die eine andere Ansichtshierarchie für `Phone` und `Tablet` Idiome erstellt.

Plattformspezifische visuelle Anpassungen online lesen: https://riptutorial.com/de/xamarin-forms/topic/5012/plattformspezifische-visuelle-anpassungen

Kapitel 31: Plattformspezifisches Verhalten

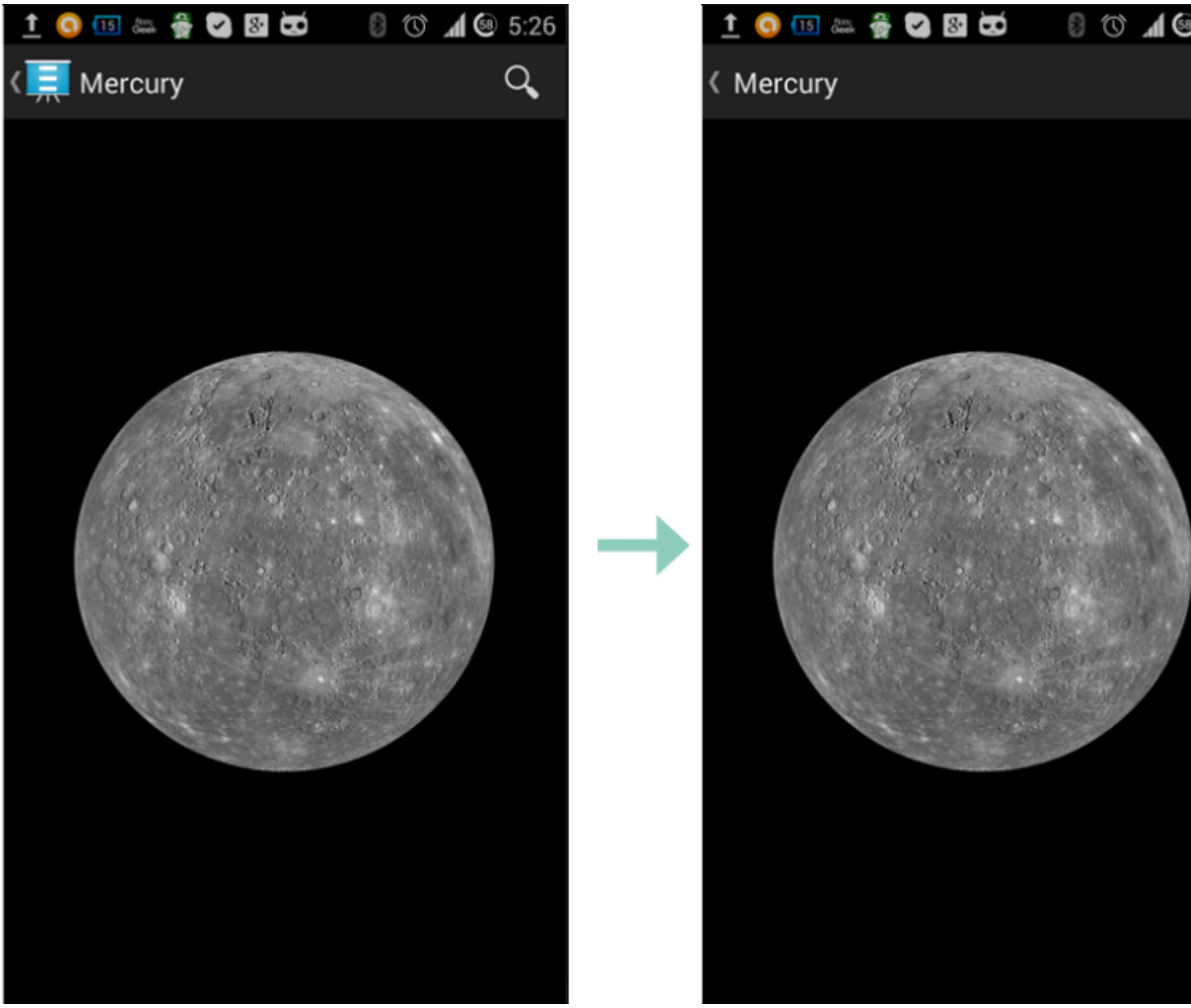
Bemerkungen

Zielplattformen

```
if(Device.OS == TargetPlatform.Android)
{

}
else if (Device.OS == TargetPlatform.iOS)
{

}
else if (Device.OS == TargetPlatform.WinPhone)
{

}
else if (Device.OS == TargetPlatform.Windows)
{

}
else if (Device.OS == TargetPlatform.Other)
{

}
```

Examples

Symbol im Navigationsheader in Anroid entfernen



Verwenden Sie ein kleines transparentes Bild namens empty.png

```
public class MyPage : ContentPage
{
    public Page()
    {
        if (Device.OS == TargetPlatform.Android)
            NavigationPage.SetTitleIcon(this, "empty.png");
    }
}
```

Verkleinern Sie die Schriftgröße des Labels in iOS

```
Label label = new Label
{
    Text = "text"
};
if(Device.OS == TargetPlatform.iOS)
{
```

```
    label.FontSize = label.FontSize - 2;
}
```

Plattformspezifisches Verhalten online lesen: https://riptutorial.com/de/xamarin-forms/topic/6636/plattformspezifisches-verhalten

Kapitel 32: SQL-Datenbank und API in Xamarin-Formularen.

Bemerkungen

Erstellen Sie Ihre eigene API mit der Microsoft SQL-Datenbank und implementieren Sie diese in einer Xamarin-Formularanwendung.

Examples

API mit SQL-Datenbank erstellen und in Xamarin-Formularen implementieren

Quellcode- Blog

SQL-Datenbank und API in Xamarin-Formularen. online lesen: https://riptutorial.com/de/xamarin-forms/topic/6513/sql-datenbank-und-api-in-xamarin-formularen-

Kapitel 33: Unit Testing

Examples

Testen der Ansichtsmodelle

Bevor wir anfangen...

In Bezug auf die Anwendungsebenen ist ViewModel eine Klasse, die alle Geschäftslogik und Regeln enthält, sodass die App den Anforderungen gemäß den Anforderungen entspricht. Es ist auch wichtig, es so unabhängig wie möglich zu machen, indem die Verweise auf Benutzeroberfläche, Datenschicht, native Funktionen und API-Aufrufe usw. reduziert werden. All dies macht Ihre VM testbar.
Kurz gesagt, Ihr ViewModel:

- Sollte nicht von UI-Klassen abhängen (Ansichten, Seiten, Stile, Ereignisse);
- Verwenden Sie keine statischen Daten einer anderen Klasse (so viel wie möglich).
- Sollte die Geschäftslogik implementieren und die Daten auf der Benutzeroberfläche vorbereiten;
- Sollte andere Komponenten (Datenbank, HTTP, UI-spezifisch) über Schnittstellen verwenden, die mithilfe von Dependency Injection aufgelöst werden.

  Ihr ViewModel verfügt möglicherweise auch über Eigenschaften anderer VMs-Typen. Für `ContactsPageViewModel` gibt es beispielsweise einen Collection-Typ wie `ObservableCollection<ContactListItemViewModel>`

Geschäftsanforderungen

Angenommen, wir haben die folgende Funktionalität, die implementiert werden muss:

```
As an unauthorized user
I want to log into the app
So that I will access the authorized features
```

Nach der Klärung der User Story haben wir folgende Szenarien definiert:

```
Scenario: trying to log in with valid non-empty creds
  Given the user is on Login screen
   When the user enters 'user' as username
    And the user enters 'pass' as password
    And the user taps the Login button
   Then the app shows the loading indicator
    And the app makes an API call for authentication

Scenario: trying to log in empty username
  Given the user is on Login screen
```

```
   When the user enters '  ' as username
    And the user enters 'pass' as password
    And the user taps the Login button
   Then the app shows an error message saying 'Please, enter correct username and password'
    And the app doesn't make an API call for authentication
```

Wir werden nur bei diesen beiden Szenarien bleiben. Natürlich sollte es viel mehr Fälle geben, und Sie sollten alle vor der eigentlichen Codierung definieren, aber es ist für uns jetzt schon genug, um uns mit dem Komponententest von Ansichtsmodellen vertraut zu machen.

Folgen wir dem klassischen TDD-Ansatz und beginnen Sie mit dem Schreiben einer leeren Klasse, die getestet wird. Dann schreiben wir Tests und machen sie grün, indem wir die Business-Funktionalität implementieren.

Gemeinsame Klassen

```
public abstract class BaseViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```

Dienstleistungen

Erinnern Sie sich, dass unser Ansichtsmodell UI- und HTTP-Klassen nicht direkt verwenden darf? Sie sollten sie stattdessen als Abstraktionen definieren und sich nicht auf Implementierungsdetails verlassen .

```
/// <summary>
/// Provides authentication functionality.
/// </summary>
public interface IAuthenticationService
{
    /// <summary>
    /// Tries to authenticate the user with the given credentials.
    /// </summary>
    /// <param name="userName">UserName</param>
    /// <param name="password">User's password</param>
    /// <returns>true if the user has been successfully authenticated</returns>
    Task<bool> Login(string userName, string password);
}

/// <summary>
/// UI-specific service providing abilities to show alert messages.
/// </summary>
public interface IAlertService
{
```

```
    /// <summary>
    /// Show an alert message to the user.
    /// </summary>
    /// <param name="title">Alert message title</param>
    /// <param name="message">Alert message text</param>
    Task ShowAlert(string title, string message);
}
```

Erstellen des ViewModel-Stubs

Ok, wir haben die Seitenklasse für den Anmeldebildschirm, aber beginnen wir zuerst mit ViewModel:

```
public class LoginPageViewModel : BaseViewModel
{
    private readonly IAuthenticationService authenticationService;
    private readonly IAlertService alertService;

    private string userName;
    private string password;
    private bool isLoading;

    private ICommand loginCommand;

    public LoginPageViewModel(IAuthenticationService authenticationService, IAlertService
alertService)
    {
        this.authenticationService = authenticationService;
        this.alertService = alertService;
    }

    public string UserName
    {
        get
        {
            return userName;
        }
        set
        {
            if (userName!= value)
            {
                userName= value;
                OnPropertyChanged();
            }
        }
    }

    public string Password
    {
        get
        {
            return password;
        }
        set
        {
            if (password != value)
            {
```

```
                password = value;
                OnPropertyChanged();
            }
        }
    }

    public bool IsLoading
    {
        get
        {
            return isLoading;
        }
        set
        {
            if (isLoading != value)
            {
                isLoading = value;
                OnPropertyChanged();
            }
        }
    }

    public ICommand LoginCommand => loginCommand ?? (loginCommand = new Command(Login));

    private void Login()
    {
        authenticationService.Login(UserName, Password);
    }
}
```

Wir haben zwei `string` Eigenschaften und einen Befehl definiert, der an die Benutzeroberfläche gebunden werden soll. Es wird nicht beschrieben, wie Sie eine Seitenklasse, ein XAML-Markup erstellen und ViewModel in diesem Thema daran binden, da sie nichts Spezifisches enthalten.

Wie erstelle ich eine LoginPageViewModel-Instanz?

Ich denke, Sie haben die VMs wahrscheinlich nur mit Konstruktor erstellt. Wie Sie sehen, ist unsere VM nun darauf angewiesen, dass 2 Services als Konstruktorparameter `var viewModel = new LoginPageViewModel()` kann `var viewModel = new LoginPageViewModel()` nicht einfach verwendet werden. Wenn Sie mit Dependency Injection nicht vertraut sind, ist es der beste Moment, um etwas darüber zu erfahren. Eine ordnungsgemäße Prüfung der Einheiten ist nicht möglich, ohne diesen Grundsatz zu kennen und zu befolgen.

Tests

Jetzt schreiben wir einige Tests gemäß den oben aufgeführten Anwendungsfällen. Zunächst müssen Sie eine neue Assembly erstellen (nur eine Klassenbibliothek oder ein spezielles Testprojekt auswählen, wenn Sie Microsoft-Komponententools verwenden möchten). Nennen Sie es etwas wie `ProjectName.Tests` und fügen Sie Ihrem ursprünglichen PCL-Projekt einen Verweis

hinzu.

In diesem Beispiel werde ich NUnit und Moq verwenden, aber Sie können mit allen Testlibs Ihrer Wahl fortfahren. Es wird nichts Besonderes mit ihnen geben.

Ok, das ist die Testklasse:

```
[TestFixture]
public class LoginPageViewModelTest
{
}
```

Tests schreiben

Hier sind die Testmethoden für die ersten beiden Szenarien. Versuchen Sie, 1 Testmethode pro 1 erwartetem Ergebnis beizubehalten und nicht alles in einem Test zu überprüfen. Dies hilft Ihnen, klarere Berichte darüber zu erhalten, was im Code fehlgeschlagen ist.

```
[TestFixture]
public class LoginPageViewModelTest
{
    private readonly Mock<IAuthenticationService> authenticationServiceMock =
        new Mock<IAuthenticationService>();
    private readonly Mock<IAlertService> alertServiceMock =
        new Mock<IAlertService>();

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_LoadingIndicatorShown(string userName, string password)
    {
        LoginPageViewModel model = CreateViewModelAndLogin(userName, password);

        Assert.IsTrue(model.IsLoading);
    }

    [TestCase("user", "pass")]
    public void LogInWithValidCreds_AuthenticationRequested(string userName, string password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(userName, password), Times.Once);
    }

    [TestCase("", "pass")]
    [TestCase("   ", "pass")]
    [TestCase(null, "pass")]
    public void LogInWithEmptyuserName_AuthenticationNotRequested(string userName, string
password)
    {
        CreateViewModelAndLogin(userName, password);

        authenticationServiceMock.Verify(x => x.Login(It.IsAny<string>(), It.IsAny<string>()),
Times.Never);
    }

    [TestCase("", "pass", "Please, enter correct username and password")]
```

```
    [TestCase("   ", "pass", "Please, enter correct username and password")]
    [TestCase(null, "pass", "Please, enter correct username and password")]
    public void LogInWithEmptyUserName_AlertMessageShown(string userName, string password,
string message)
    {
        CreateViewModelAndLogin(userName, password);

        alertServiceMock.Verify(x => x.ShowAlert(It.IsAny<string>(), message));
    }

    private LoginPageViewModel CreateViewModelAndLogin(string userName, string password)
    {
        var model = new LoginPageViewModel(
            authenticationServiceMock.Object,
            alertServiceMock.Object);

        model.UserName = userName;
        model.Password = password;

        model.LoginCommand.Execute(null);

        return model;
    }
}
```

Und es geht los:

Nun ist es das Ziel, die korrekte Implementierung für die `Login` Methode von ViewModel zu schreiben, und das war's.

Implementierung der Geschäftslogik

```
private async void Login()
{
    if (String.IsNullOrWhiteSpace(UserName) || String.IsNullOrWhiteSpace(Password))
    {
        await alertService.ShowAlert("Warning", "Please, enter correct username and
password");
    }
    else
```

```
    {
        IsLoading = true;
        bool isAuthenticated = await authenticationService.Login(UserName, Password);
    }
}
```

Und nachdem die Tests erneut ausgeführt wurden:

Jetzt können Sie Ihren Code weiterhin mit neuen Tests abdecken, um ihn stabiler und regressionssicherer zu machen.

Unit Testing online lesen: https://riptutorial.com/de/xamarin-forms/topic/3529/unit-testing

Kapitel 34: Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet?

Bemerkungen

Weitere Informationen finden Sie in der offiziellen Dokumentation zu Xamarin Forms:

https://www.xamarin.com/forms

Examples

Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet?

Xamarin wird immer beliebter - es ist schwer zu entscheiden, wann Xamarin.Forms und wann Xamarin.Platform verwendet wird (also Xamarin.iOS und Xamarin.Android).

Zunächst sollten Sie wissen, für welche Art von Anwendungen Sie Xamarin.Forms verwenden können:

1. Prototypen - um zu visualisieren, wie Ihre Anwendung auf den verschiedenen Geräten aussehen wird.

2. Anwendungen, die keine plattformspezifischen Funktionen (wie APIs) erfordern - hier ist jedoch zu beachten, dass Xamarin eifrig daran arbeitet, so viele plattformübergreifende Kompatibilität wie möglich bereitzustellen.

3. Anwendungen, bei denen Code-Sharing von entscheidender Bedeutung ist - wichtiger als die Benutzeroberfläche.

4. Anwendungen, bei denen Daten angezeigt werden, sind wichtiger als erweiterte Funktionen

Es gibt auch viele andere Faktoren:

1. Wer ist für die Anwendungsentwicklung verantwortlich? Wenn Ihr Team aus erfahrenen mobilen Entwicklern besteht, können Sie mit Xamarin.Forms problemlos umgehen. Wenn Sie jedoch einen Entwickler pro Plattform haben (native Entwicklung), können Formulare eine größere Herausforderung darstellen.

2. Bitte beachten Sie auch, dass Sie mit Xamarin.Forms manchmal noch auf einige Probleme stoßen können - die Plattform Xamarin.Forms wird immer noch verbessert.

3. Eine schnelle Entwicklung ist manchmal sehr wichtig - um Kosten und Zeit zu reduzieren, können Sie Formulare verwenden.

4. Bei der Entwicklung von Unternehmensanwendungen ohne erweiterte Funktionen ist es besser, Xamarin.Forms zu verwenden. Damit können Sie Moduscode nicht im mobilen

Bereich, sondern allgemein freigeben. Einige Teile des Codes können auf vielen Plattformen gemeinsam genutzt werden.

Sie sollten Xamarin.Forms nicht verwenden, wenn:

1. Sie müssen benutzerdefinierte Funktionen erstellen und auf plattformspezifische APIs zugreifen
2. Sie müssen eine benutzerdefinierte Benutzeroberfläche für die mobile Anwendung erstellen
3. Wenn einige Funktionen für Xamarin.Forms nicht bereit sind (z. B. ein bestimmtes Verhalten auf dem mobilen Gerät)
4. Ihr Team besteht aus plattformspezifischen mobilen Entwicklern (mobile Entwicklung in Java und / oder Swift / Objective C)

Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet? online lesen: https://riptutorial.com/de/xamarin-forms/topic/6869/warum-xamarin-formulare-und-wann-werden-xamarin-formulare-verwendet-

Kapitel 35: Xamarin Plugin

Examples

Plugin freigeben

Einfache Möglichkeit, Nachrichten oder Links zu teilen, Text in die Zwischenablage zu kopieren oder einen Browser in einer Xamarin- oder Windows-App zu öffnen.

Verfügbar bei NuGet: https://www.nuget.org/packages/Plugin.Share/

XAML

```
<StackLayout Padding="20" Spacing="20">
      <Button StyleId="Text" Text="Share Text" Clicked="Button_OnClicked"/>
      <Button StyleId="Link" Text="Share Link" Clicked="Button_OnClicked"/>
      <Button StyleId="Browser" Text="Open Browser" Clicked="Button_OnClicked"/>
      <Label Text=""/>

 </StackLayout>
```

C #

```
 async void Button_OnClicked(object sender, EventArgs e)
        {
            switch (((Button)sender).StyleId)
            {
                case "Text":
                    await CrossShare.Current.Share("Follow @JamesMontemagno on Twitter",
"Share");
                    break;
                case "Link":
                    await CrossShare.Current.ShareLink("http://motzcod.es", "Checkout my
blog", "MotzCod.es");
                    break;
                case "Browser":
                    await CrossShare.Current.OpenBrowser("http://motzcod.es");
                    break;
            }
        }
```

ExternalMaps

Externes Karten-Plugin Öffnen Sie externe Karten, um zu einem bestimmten Ort oder einer bestimmten Adresse zu navigieren. Startmöglichkeit mit Navigationsoption auch für iOS.

Verfügbar bei NuGet: [https://www.nuget.org/packages/Xam.Plugin.ExternalMaps/([1]

XAML

```
  <StackLayout Spacing="10" Padding="10">
```

```
      <Button x:Name="navigateAddress" Text="Navigate to Address"/>
      <Button x:Name="navigateLatLong" Text="Navigate to Lat|Long"/>
      <Label Text=""/>

    </StackLayout>
```

Code

```
namespace PluginDemo
{
    public partial class ExternalMaps : ContentPage
    {
        public ExternalMaps()
        {
            InitializeComponent();
            navigateLatLong.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Space Needle", 47.6204, -122.3491);
            };

            navigateAddress.Clicked += (sender, args) =>
            {
                CrossExternalMaps.Current.NavigateTo("Xamarin", "394 pacific ave.", "San
Francisco", "CA", "94111", "USA", "USA");
            };
        }
    }
}
```

Geolocator Plugin

Greifen Sie einfach auf die Geolokalisierung in Xamarin.iOS, Xamarin.Android und Windows zu.

Verfügbares Nuget: [https://www.nuget.org/packages/Xam.Plugin.Geolocator/(

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Button x:Name="buttonGetGPS" Text="Get GPS"/>
      <Label x:Name="labelGPS"/>
      <Button x:Name="buttonTrack" Text="Track Movements"/>
      <Label x:Name="labelGPSTrack"/>
      <Label Text=""/>

    </StackLayout>
```

Code

```
namespace PluginDemo
{
    public partial class GeolocatorPage : ContentPage
    {
        public GeolocatorPage()
        {
            InitializeComponent();
            buttonGetGPS.Clicked += async (sender, args) =>
```

```
            {
                try
                {
                    var locator = CrossGeolocator.Current;
                    locator.DesiredAccuracy = 1000;
                    labelGPS.Text = "Getting gps";

                    var position = await locator.GetPositionAsync(timeoutMilliseconds: 10000);

                    if (position == null)
                    {
                        labelGPS.Text = "null gps :(";
                        return;
                    }
                    labelGPS.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2}
\nAltitude: {3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                        position.Timestamp, position.Latitude, position.Longitude,
                        position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            buttonTrack.Clicked += async (object sender, EventArgs e) =>
            {
                try
                {
                    if (CrossGeolocator.Current.IsListening)
                    {
                        await CrossGeolocator.Current.StopListeningAsync();
                        labelGPSTrack.Text = "Stopped tracking";
                        buttonTrack.Text = "Stop Tracking";
                    }
                    else
                    {
                        if (await CrossGeolocator.Current.StartListeningAsync(30000, 0))
                        {
                            labelGPSTrack.Text = "Started tracking";
                            buttonTrack.Text = "Track Movements";
                        }
                    }
                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }

        protected override void OnAppearing()
        {
            base.OnAppearing();
            try
```

```
            {
                CrossGeolocator.Current.PositionChanged +=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError +=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }

        void CrossGeolocator_Current_PositionError(object sender,
Plugin.Geolocator.Abstractions.PositionErrorEventArgs e)
        {

            labelGPSTrack.Text = "Location error: " + e.Error.ToString();
        }

        void CrossGeolocator_Current_PositionChanged(object sender,
Plugin.Geolocator.Abstractions.PositionEventArgs e)
        {
            var position = e.Position;
            labelGPSTrack.Text = string.Format("Time: {0} \nLat: {1} \nLong: {2} \nAltitude:
{3} \nAltitude Accuracy: {4} \nAccuracy: {5} \nHeading: {6} \nSpeed: {7}",
                position.Timestamp, position.Latitude, position.Longitude,
                position.Altitude, position.AltitudeAccuracy, position.Accuracy,
position.Heading, position.Speed);


        }

        protected override void OnDisappearing()
        {
            base.OnDisappearing();
            try
            {
                CrossGeolocator.Current.PositionChanged -=
CrossGeolocator_Current_PositionChanged;
                CrossGeolocator.Current.PositionError -=
CrossGeolocator_Current_PositionError;
            }
            catch
            {
            }
        }
    }
}
```

Medien Plugin

Machen Sie Fotos und Videos aus einer plattformübergreifenden API.

Verfügbares Nuget: [https://www.nuget.org/packages/Xam.Plugin.Media/([1]

XAML

```
<StackLayout Spacing="10" Padding="10">
      <Button x:Name="takePhoto" Text="Take Photo"/>
```

```
      <Button x:Name="pickPhoto" Text="Pick Photo"/>
      <Button x:Name="takeVideo" Text="Take Video"/>
      <Button x:Name="pickVideo" Text="Pick Video"/>
      <Label Text="Save to Gallery"/>
      <Switch x:Name="saveToGallery" IsToggled="false" HorizontalOptions="Center"/>
      <Label Text="Image will show here"/>
      <Image x:Name="image"/>
      <Label Text=""/>

    </StackLayout>
```

Code

```
namespace PluginDemo
{
    public partial class MediaPage : ContentPage
    {
        public MediaPage()
        {
            InitializeComponent();
            takePhoto.Clicked += async (sender, args) =>
            {

                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakePhotoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.TakePhotoAsync(new
Plugin.Media.Abstractions.StoreCameraMediaOptions
                    {
                        Directory = "Sample",
                        Name = "test.jpg",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("File Location", (saveToGallery.IsToggled ?
file.AlbumPath : file.Path), "OK");

                    image.Source = ImageSource.FromStream(() =>
                    {
                        var stream = file.GetStream();
                        file.Dispose();
                        return stream;
                    });
                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
```

```
            pickPhoto.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickPhotoSupported)
                {
                    await DisplayAlert("Photos Not Supported", ":( Permission not granted to
photos.", "OK");
                    return;
                }
                try
                {
                    Stream stream = null;
                    var file = await CrossMedia.Current.PickPhotoAsync().ConfigureAwait(true);


                    if (file == null)
                        return;

                    stream = file.GetStream();
                    file.Dispose();

                    image.Source = ImageSource.FromStream(() => stream);

                }
                catch //(Exception ex)
                {
                   // Xamarin.Insights.Report(ex);
                   // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            takeVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsCameraAvailable ||
!CrossMedia.Current.IsTakeVideoSupported)
                {
                    await DisplayAlert("No Camera", ":( No camera avaialble.", "OK");
                    return;
                }

                try
                {
                    var file = await CrossMedia.Current.TakeVideoAsync(new
Plugin.Media.Abstractions.StoreVideoOptions
                    {
                        Name = "video.mp4",
                        Directory = "DefaultVideos",
                        SaveToAlbum = saveToGallery.IsToggled
                    });

                    if (file == null)
                        return;

                    await DisplayAlert("Video Recorded", "Location: " +
(saveToGallery.IsToggled ? file.AlbumPath : file.Path), "OK");

                    file.Dispose();

                }
                catch //(Exception ex)
                {
```

```
                    // Xamarin.Insights.Report(ex);
                    // await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };

            pickVideo.Clicked += async (sender, args) =>
            {
                if (!CrossMedia.Current.IsPickVideoSupported)
                {
                    await DisplayAlert("Videos Not Supported", ":( Permission not granted to
videos.", "OK");
                    return;
                }
                try
                {
                    var file = await CrossMedia.Current.PickVideoAsync();

                    if (file == null)
                        return;

                    await DisplayAlert("Video Selected", "Location: " + file.Path, "OK");
                    file.Dispose();

                }
                catch //(Exception ex)
                {
                    //Xamarin.Insights.Report(ex);
                    //await DisplayAlert("Uh oh", "Something went wrong, but don't worry we
captured it in Xamarin Insights! Thanks.", "OK");
                }
            };
        }
    }
}
```

Messaging-Plugin

Messaging-Plugin für Xamarin und Windows, um einen Anruf zu tätigen, eine SMS zu senden oder eine E-Mail mit den Standard-Messaging-Anwendungen der verschiedenen mobilen Plattformen zu senden

Verfügbares Nuget: [https://www.nuget.org/packages/Xam.Plugins.Messaging/([1]

XAML

```
 <StackLayout Spacing="10" Padding="10">
      <Entry Placeholder="Phone Number" x:Name="phone"/>
      <Button x:Name="buttonSms" Text="Send SMS"/>
      <Button x:Name="buttonCall" Text="Call Phone Number"/>
      <Entry Placeholder="E-mail Address" x:Name="email"/>
      <Button x:Name="buttonEmail" Text="Send E-mail"/>
      <Label Text=""/>

    </StackLayout>
```

Code

```
namespace PluginDemo
{
    public partial class MessagingPage : ContentPage
    {
        public MessagingPage()
        {
            InitializeComponent();
            buttonCall.Clicked += async (sender, e) =>
            {
                try
                {
                    // Make Phone Call
                    var phoneCallTask = MessagingPlugin.PhoneDialer;
                    if (phoneCallTask.CanMakePhoneCall)
                        phoneCallTask.MakePhoneCall(phone.Text);
                    else
                        await DisplayAlert("Error", "This device can't place calls", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonSms.Clicked += async (sender, e) =>
            {
                try
                {

                    var smsTask = MessagingPlugin.SmsMessenger;
                    if (smsTask.CanSendSms)
                        smsTask.SendSms(phone.Text, "Hello World");
                    else
                        await DisplayAlert("Error", "This device can't send sms", "OK");
                }
                catch
                {
                   // await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };

            buttonEmail.Clicked += async (sender, e) =>
            {
                try
                {
                    var emailTask = MessagingPlugin.EmailMessenger;
                    if (emailTask.CanSendEmail)
                        emailTask.SendEmail(email.Text, "Hello there!", "This was sent from
the Xamrain Messaging Plugin from shared code!");
                    else
                        await DisplayAlert("Error", "This device can't send emails", "OK");
                }
                catch
                {
//await DisplayAlert("Error", "Unable to perform action", "OK");
                }
            };
        }
    }
}
```

Plugins für Berechtigungen

Überprüfen Sie, ob Ihre Benutzer Berechtigungen für allgemeine Berechtigungsgruppen für iOS und Android erteilt oder verweigert haben.

Darüber hinaus können Sie Berechtigungen mit einer einfachen, plattformübergreifenden async / awaitified-API anfordern.

Available Nuget: https://www.nuget.org/packages/Plugin.Permissions Link-Beschreibung hier eingeben **XAML**

XAML

```
 <StackLayout Padding="30" Spacing="10">
      <Button Text="Get Location" Clicked="Button_OnClicked"></Button>
      <Label x:Name="LabelGeolocation"></Label>
      <Button Text="Calendar" StyleId="Calendar"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Camera" StyleId="Camera" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Contacts" StyleId="Contacts"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Microphone" StyleId="Microphone"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Phone" StyleId="Phone" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Photos" StyleId="Photos" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Reminders" StyleId="Reminders"
Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sensors" StyleId="Sensors" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Sms" StyleId="Sms" Clicked="ButtonPermission_OnClicked"></Button>
      <Button Text="Storage" StyleId="Storage" Clicked="ButtonPermission_OnClicked"></Button>
      <Label Text=""/>

    </StackLayout>
```

Code

```
bool busy;
        async void ButtonPermission_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            var status = PermissionStatus.Unknown;
            switch (((Button)sender).StyleId)
            {
                case "Calendar":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Calendar);
                    break;
                case "Camera":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Camera);
                    break;
```

```
                case "Contacts":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Contacts);
                    break;
                case "Microphone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Microphone);
                    break;
                case "Phone":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Phone);
                    break;
                case "Photos":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Photos);
                    break;
                case "Reminders":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Reminders);
                    break;
                case "Sensors":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sensors);
                    break;
                case "Sms":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Sms);
                    break;
                case "Storage":
                    status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Storage);
                    break;
            }

            await DisplayAlert("Results", status.ToString(), "OK");

            if (status != PermissionStatus.Granted)
            {
                switch (((Button)sender).StyleId)
                {
                    case "Calendar":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Calendar))[Permission.Calendar];
                        break;
                    case "Camera":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Camera))[Permission.Camera];
                        break;
                    case "Contacts":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Contacts))[Permission.Contacts];
                        break;
                    case "Microphone":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Microphone))[Permission.Microphone];

                        break;
                    case "Phone":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Phone))[Permission.Phone];
                        break;
```

```
                    case "Photos":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Photos))[Permission.Photos];
                        break;
                    case "Reminders":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Reminders))[Permission.Reminders];
                        break;
                    case "Sensors":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sensors))[Permission.Sensors];
                        break;
                    case "Sms":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Sms))[Permission.Sms];
                        break;
                    case "Storage":
                        status = (await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Storage))[Permission.Storage];
                        break;
                }

                await DisplayAlert("Results", status.ToString(), "OK");

            }

            busy = false;
            ((Button)sender).IsEnabled = true;
        }

        async void Button_OnClicked(object sender, EventArgs e)
        {
            if (busy)
                return;

            busy = true;
            ((Button)sender).IsEnabled = false;

            try
            {
                var status = await
CrossPermissions.Current.CheckPermissionStatusAsync(Permission.Location);
                if (status != PermissionStatus.Granted)
                {
                    if (await
CrossPermissions.Current.ShouldShowRequestPermissionRationaleAsync(Permission.Location))
                    {
                        await DisplayAlert("Need location", "Gunna need that location", "OK");
                    }

                    var results = await
CrossPermissions.Current.RequestPermissionsAsync(Permission.Location);
                    status = results[Permission.Location];
                }

                if (status == PermissionStatus.Granted)
                {
                    var results = await CrossGeolocator.Current.GetPositionAsync(10000);
                    LabelGeolocation.Text = "Lat: " + results.Latitude + " Long: " +
results.Longitude;
                }
```

```
                else if (status != PermissionStatus.Unknown)
                {
                    await DisplayAlert("Location Denied", "Can not continue, try again.",
"OK");
                }
            }
            catch (Exception ex)
            {

                LabelGeolocation.Text = "Error: " + ex;
            }

            ((Button)sender).IsEnabled = true;
            busy = false;
        }
```

Xamarin Plugin online lesen: https://riptutorial.com/de/xamarin-forms/topic/7017/xamarin-plugin

Kapitel 36: Xamarin Relatives Layout

Bemerkungen

Die Verwendung von *ForceLayout* in diesem Fall

Die Größe von Beschriftungen und Schaltflächen ändert sich je nach Text. Wenn die untergeordneten Elemente zum Layout hinzugefügt werden, bleibt ihre Größe in Breite und Höhe gleich 0. Zum Beispiel:

```
relativeLayout.Children.Add(label,
    Constraint.RelativeToParent(parent => label.Width));
```

Der obige Ausdruck gibt 0 zurück, da width momentan 0 ist. Um dies zu *umgehen* , müssen wir auf das *SizeChanged*- Ereignis *achten* und wenn sich die Größe ändert, sollten wir das Layout erzwingen, um es neu zu zeichnen.

```
label.SizeChanged += (s, e) => relativeLayout.ForceLayout();
```

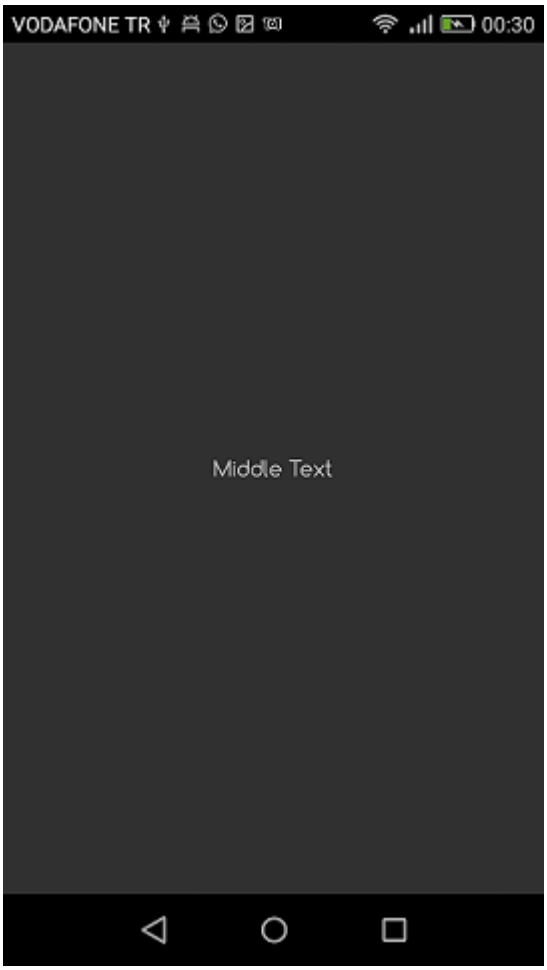
Für eine Ansicht wie *BoxView ist* dies nicht *erforderlich* . Weil wir ihre Größe bei der Instantiierung definieren können. Das andere ist, in beiden Fällen können wir deren Breite und Höhe als Einschränkung definieren, wenn wir sie zum Layout hinzufügen. Zum Beispiel:

```
relativeLayout.Children.Add(label,
    Constraint.Constant(0),
    Constraint.Constant(0),
    //Width constraint
    Constraint.Constant(30),
    //Height constraint
    Constraint.Constant(40));
```

Dadurch wird die Beschriftung an den Punkt 0, 0 angehängt. Breite und Höhe der Beschriftung betragen 30 und 40. Wenn der Text jedoch zu lang ist, werden einige der Beschriftungen möglicherweise nicht angezeigt. Wenn Ihr Etikett eine hohe Höhe hat oder haben kann, können Sie die *LineBreakMode*- Eigenschaft des Etiketts verwenden. Was kann den Text einschließen. Es gibt viele Optionen in *LineBreakMode* .

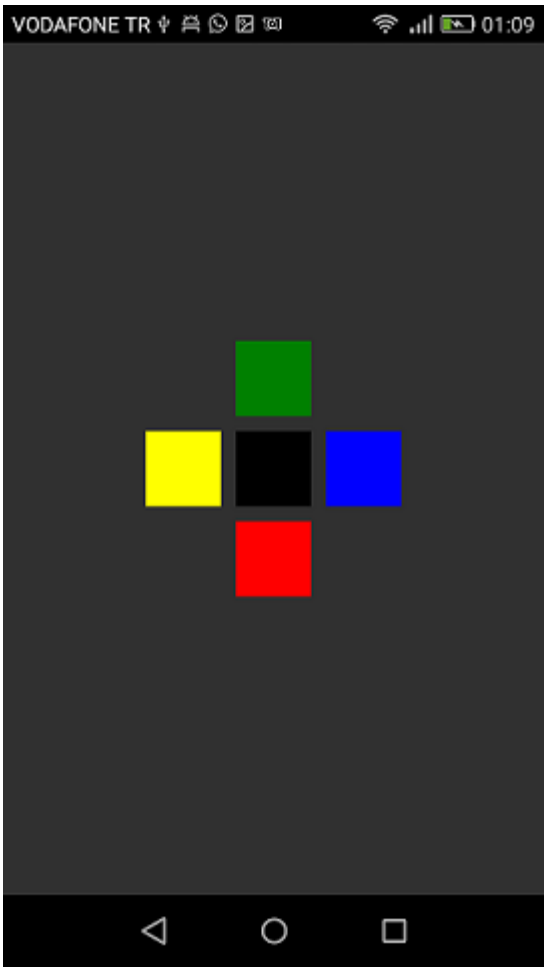
Examples

Seite mit einem einfachen Etikett in der Mitte




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;
    Label MiddleText;

    public MyPage()
    {
        _layout = new RelativeLayout();

        MiddleText = new Label
        {
            Text = "Middle Text"
        };

        MiddleText.SizeChanged += (s, e) =>
        {
            //We will force the layout so it will know the actual width and height of the
label
            //Otherwise width and height of the label remains 0 as far as layout knows
            _layout.ForceLayout();
        };

        _layout.Children.Add(MiddleText
            Constraint.RelativeToParent(parent => parent.Width / 2 - MiddleText.Width / 2),
            Constraint.RelativeToParent(parent => parent.Height / 2 - MiddleText.Height / 2));

        Content = _layout;
    }
}
```

Kiste für Kiste




```
public class MyPage : ContentPage
{
    RelativeLayout _layout;

    BoxView centerBox;
    BoxView rightBox;
    BoxView leftBox;
    BoxView topBox;
    BoxView bottomBox;

    const int spacing = 10;
    const int boxSize = 50;

    public MyPage()
    {
        _layout = new RelativeLayout();

        centerBox = new BoxView
        {
            BackgroundColor = Color.Black
        };

        rightBox = new BoxView
        {
            BackgroundColor = Color.Blue,
            //You can both set width and hight here
            //Or when adding the control to the layout
```

```
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        leftBox = new BoxView
        {
            BackgroundColor = Color.Yellow,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        topBox = new BoxView
        {
            BackgroundColor = Color.Green,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        bottomBox = new BoxView
        {
            BackgroundColor = Color.Red,
            WidthRequest = boxSize,
            HeightRequest = boxSize
        };

        //First adding center box since other boxes will be relative to center box
        _layout.Children.Add(centerBox,
            //Constraint for X, centering it horizontally
            //We give the expression as a paramater, parent is our layout in this case
            Constraint.RelativeToParent(parent => parent.Width / 2 - boxSize / 2),
            //Constraint for Y, centering it vertically
            Constraint.RelativeToParent(parent => parent.Height / 2 - boxSize / 2),
            //Constraint for Width
            Constraint.Constant(boxSize),
            //Constraint for Height
            Constraint.Constant(boxSize));

        _layout.Children.Add(leftBox,
            //The x constraint will relate on some level to centerBox
            //Which is the first parameter in this case
            //We both need to have parent and centerBox, which will be called sibling,
            //in our expression paramters
            //This expression will be our second paramater
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X - spacing -
boxSize),
            //Since we only need to move it left,
            //it's Y constraint will be centerBox' position at Y axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
            //No need to define the size constraints
            //Since we initialize them during instantiation
        );

        _layout.Children.Add(rightBox,
            //The only difference hear is adding spacing and boxSize instead of substracting
them
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X + spacing +
boxSize),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y)
        );

        _layout.Children.Add(topBox,
```

```
            //Since we are going to move it vertically this thime
            //We need to do the math on Y Constraint
            //In this case, X constraint will be centerBox' position at X axis
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            //We will do the math on Y axis this time
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y  - spacing -
boxSize)
        );

        _layout.Children.Add(bottomBox,
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.X),
            Constraint.RelativeToView(centerBox, (parent, sibling) => sibling.Y + spacing +
boxSize)
        );

        Content = _layout;
    }
}
```

Xamarin Relatives Layout online lesen: https://riptutorial.com/de/xamarin-forms/topic/6583/xamarin-relatives-layout

Kapitel 37: Xamarin.Forms Cells

Examples

EntryCell

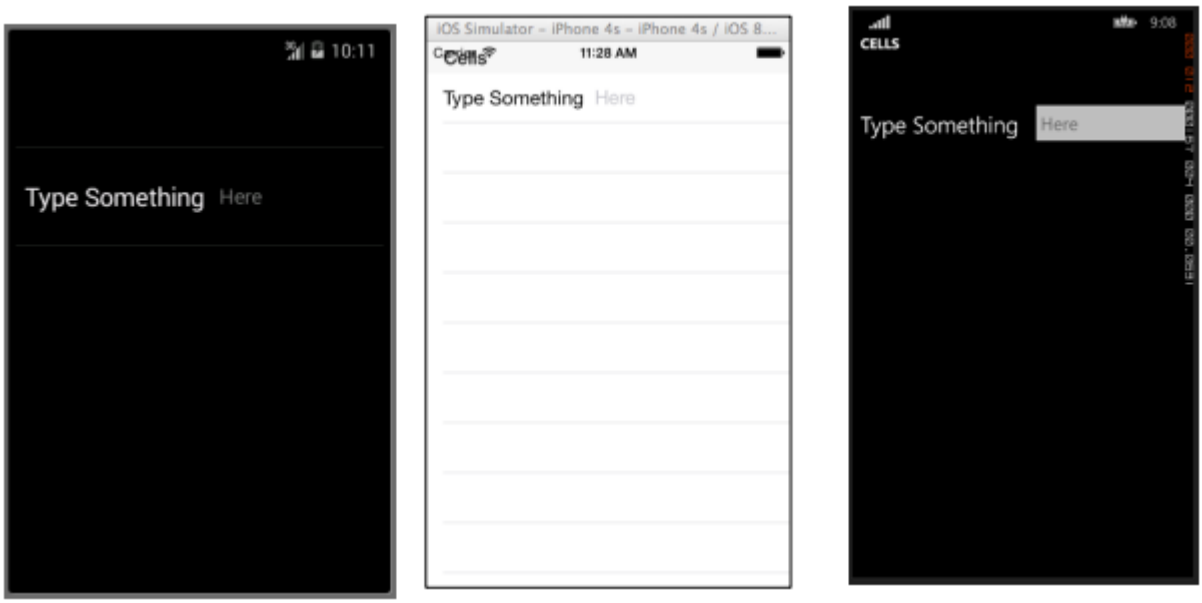
Eine EntryCell ist eine Zelle, die die Fähigkeiten eines Labels und eines Eintrags kombiniert. Die EntryCell kann in Szenarien hilfreich sein, wenn Sie in Ihrer Anwendung einige Funktionen zum Sammeln von Daten vom Benutzer erstellen. Sie können leicht in eine TableView eingefügt und als einfaches Formular behandelt werden.

XAML

```
<EntryCell Label="Type Something"
Placeholder="Here"/>
```

Code

```
var entryCell = new EntryCell {
Label = "Type Something",
Placeholder = "Here"
};
```



SwitchCell

Eine SwitchCell ist eine Zelle, die die Funktionen eines Labels und eines Ein-Aus-Schalters kombiniert. Eine SwitchCell kann hilfreich sein, um die Funktionalität oder sogar Benutzereinstellungen oder Konfigurationsoptionen ein- und auszuschalten.

XAML

```
<SwitchCell Text="Switch It Up!" />
```

Code

```
var switchCell = new SwitchCell {
Text = "Switch It Up!"
};
```

TextCell

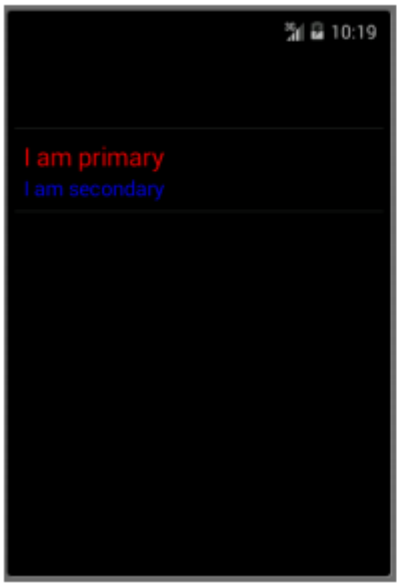
Eine TextCell ist eine Zelle mit zwei separaten Textbereichen zum Anzeigen von Daten. Eine TextCell wird normalerweise zu Informationszwecken in TableView- und ListView-Steuerelementen verwendet. Die beiden Textbereiche sind vertikal ausgerichtet, um den Platz innerhalb der Zelle zu maximieren. Diese Art von Zelle wird häufig auch zur Anzeige hierarchischer Daten verwendet. Wenn der Benutzer diese Zelle antippt, navigiert er zu einer anderen Seite.

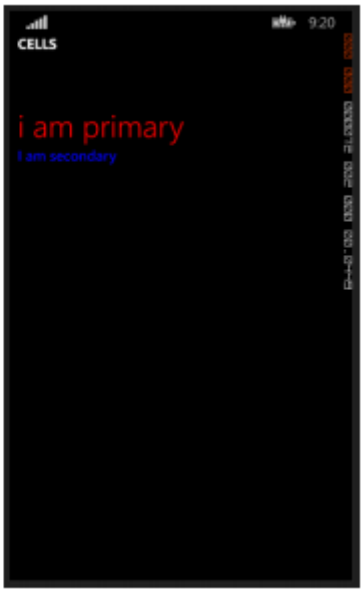
XAML

```
<TextCell Text="I am primary"
TextColor="Red"
Detail="I am secondary"
DetailColor="Blue"/>
```

Code

```
var textCell = new TextCell {
Text = "I am primary",
TextColor = Color.Red,
Detail = "I am secondary",
DetailColor = Color.Blue
};
```





ImageCell

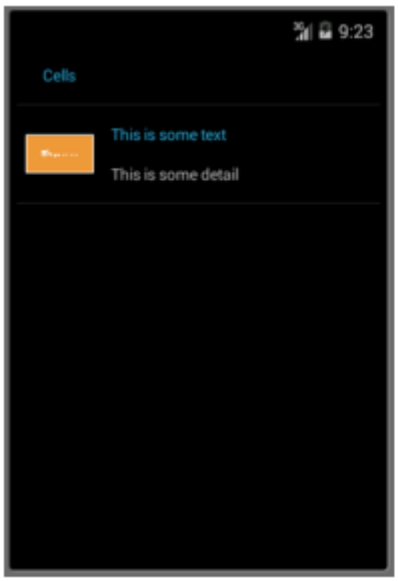
Eine ImageCell ist genau so, wie es sich anhört. Es ist eine einfache Zelle, die nur ein Bild enthält. Diese Steuerung funktioniert sehr ähnlich wie eine normale Bildsteuerung, jedoch mit weitaus weniger Glocken.

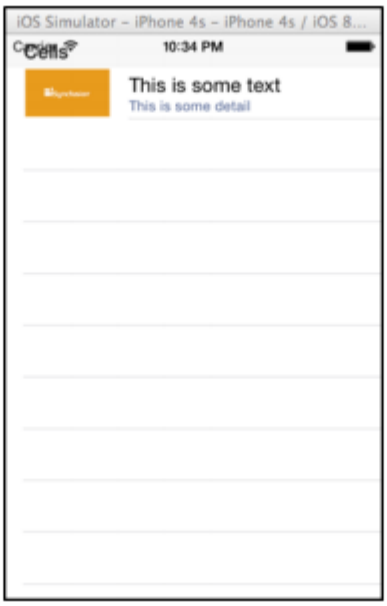
XAML

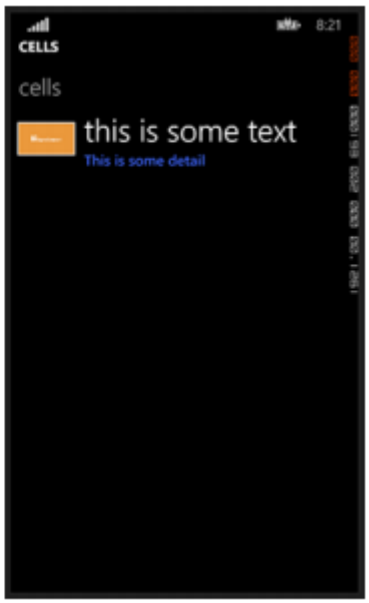
```
<ImageCell ImageSource="http://d2g29cya9iq7ip.cloudfront.net/content/imag
es/company/aboutus-video-bg.png?v=25072014072745")),
 Text="This is some text"
 Detail="This is some detail" />
```

Code

```
var imageCell = new ImageCell {
ImageSource = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.clou
109
dfront.net/content/images/company/aboutus-videobg.png?v=25072014072745")),
 Text = "This is some text",
 Detail = "This is some detail"
};
```







ViewCell

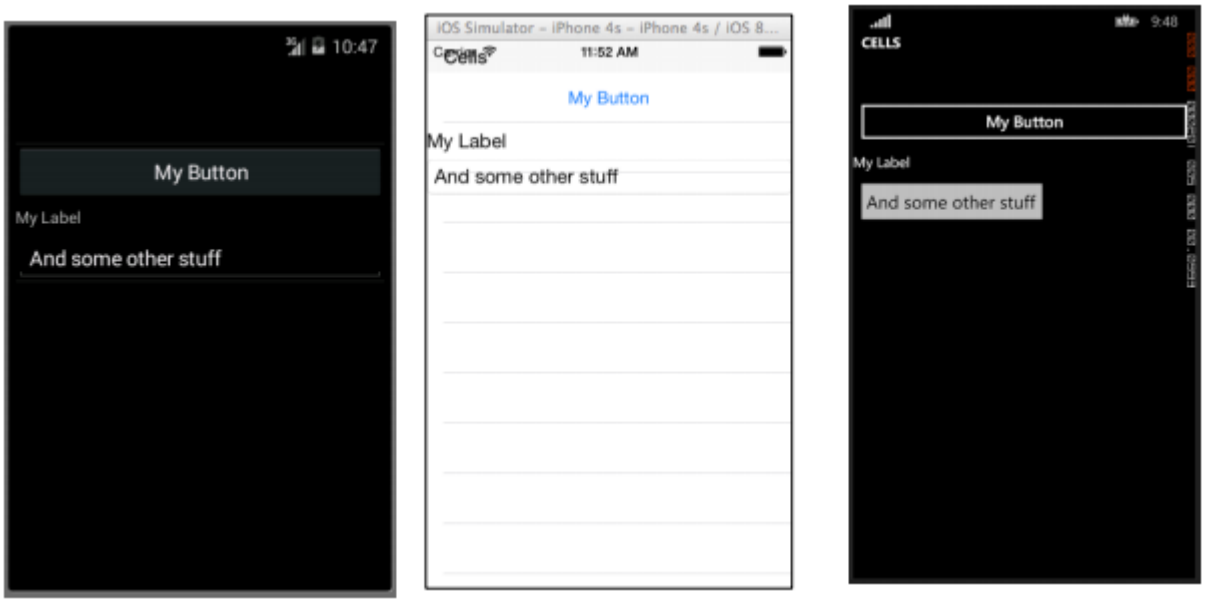
Sie können eine ViewCell als leere Tafel betrachten. Es ist Ihre persönliche Leinwand, um eine Zelle zu erstellen, die genau so aussieht, wie Sie es möchten. Sie können es sogar aus Instanzen mehrerer anderer View-Objekte zusammenstellen, die mit Layout-Steuerelementen zusammengefügt werden. Sie sind nur durch Ihre Vorstellungskraft begrenzt. Und vielleicht Bildschirmgröße.

XAML

```
<ViewCell>
<ViewCell.View>
<StackLayout>
<Button Text="My Button"/>

<Label Text="My Label"/>
<Entry Text="And some other stuff"/>
</StackLayout>
</ViewCell.View>
</ViewCell>
```

Code

```
var button = new Button { Text = "My Button" };
var label = new Label { Text = "My Label" };
var entry = new Entry { Text ="And some other stuff" };
var viewCell = new ViewCell {
View = new StackLayout {
Children = { button, label, entry }
}
};
```



Xamarin.Forms Cells online lesen: https://riptutorial.com/de/xamarin-forms/topic/7370/xamarin-forms-cells

Kapitel 38: Xamarin.Forms Seite

Examples

TabbedPage

Eine TabbedPage ähnelt einer NavigationPage insofern, als sie eine einfache Navigation zwischen mehreren untergeordneten Seitenobjekten ermöglicht und verwaltet. Der Unterschied ist, dass im Allgemeinen auf jeder Plattform oben oder unten auf dem Bildschirm eine Art Leiste angezeigt wird, die die meisten, wenn nicht alle verfügbaren untergeordneten Seitenobjekte anzeigt. In Xamarin.Forms-Anwendungen ist eine TabbedPage im Allgemeinen hilfreich, wenn Sie über eine kleine vordefinierte Anzahl von Seiten verfügen, zwischen denen Benutzer navigieren können, beispielsweise ein Menü oder ein einfacher Assistent, der am oberen oder unteren Bildschirmrand positioniert werden kann.

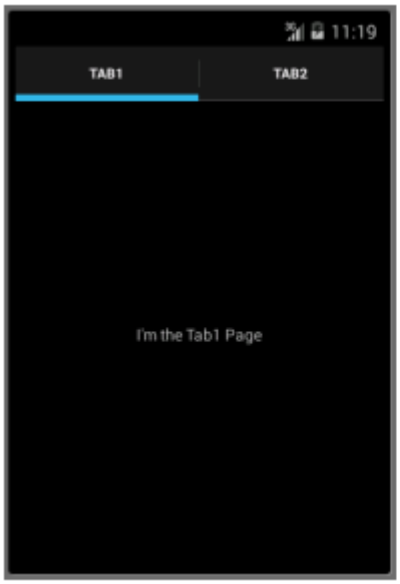
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="XamlBasics.SampleXaml">
  <TabbedPage.Children>
    <ContentPage Title="Tab1">
      <Label Text="I'm the Tab1 Page"
             HorizontalOptions="Center"
             VerticalOptions="Center"/>
    </ContentPage>
    <ContentPage Title="Tab2">
    <Label Text="I'm the Tab2 Page"
           HorizontalOptions="Center"
           VerticalOptions="Center"/>
    </ContentPage>
  </TabbedPage.Children>
</TabbedPage>
```

Code

```
var page1 = new ContentPage {
Title = "Tab1",
Content = new Label {
Text = "I'm the Tab1 Page",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
};
var page2 = new ContentPage {
Title = "Tab2",
Content = new Label {
Text = "I'm the Tab2 Page",
HorizontalOptions = LayoutOptions.Center,
66
```

```
VerticalOptions = LayoutOptions.Center
}
};
var tabbedPage = new TabbedPage {
Children = { page1, page2 }
};
```



Inhaltsseite

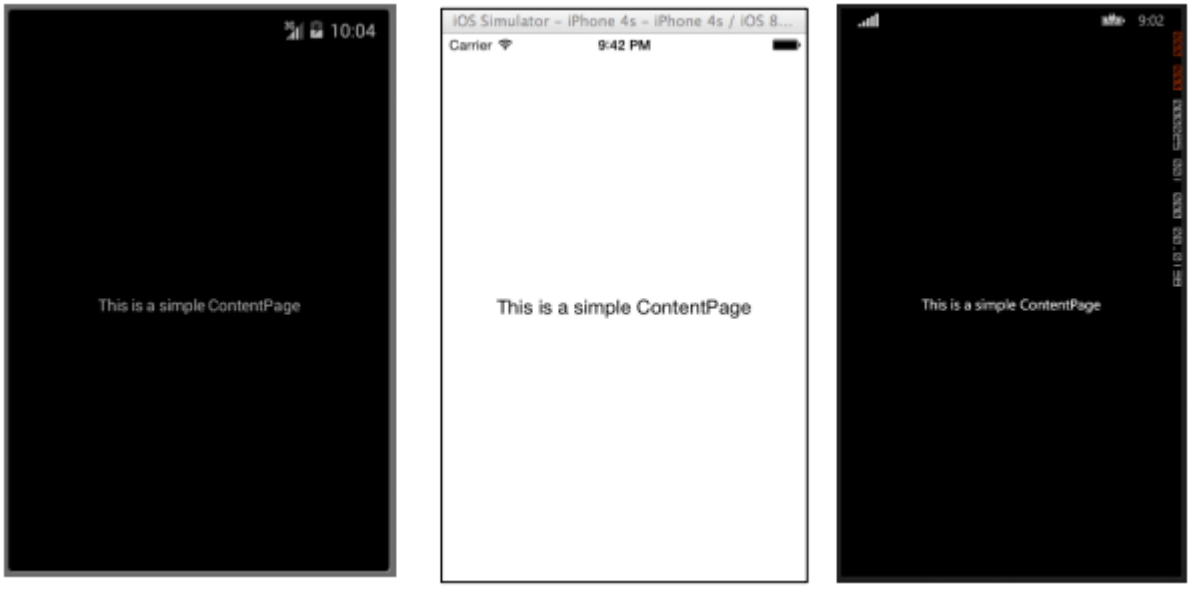
ContentPage: Zeigt eine einzelne Ansicht an.

XAML

```
<?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="XamlBasics.SampleXaml">
<Label Text="This is a simple ContentPage"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
```

Code

```
var label = new Label {
Text = "This is a simple ContentPage",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
};
var contentPage = new ContentPage {
Content = label
};
```



MasterDetailPage

MasterDetailPage: Verwaltet zwei separate Seiten (Fenster) mit Informationen.

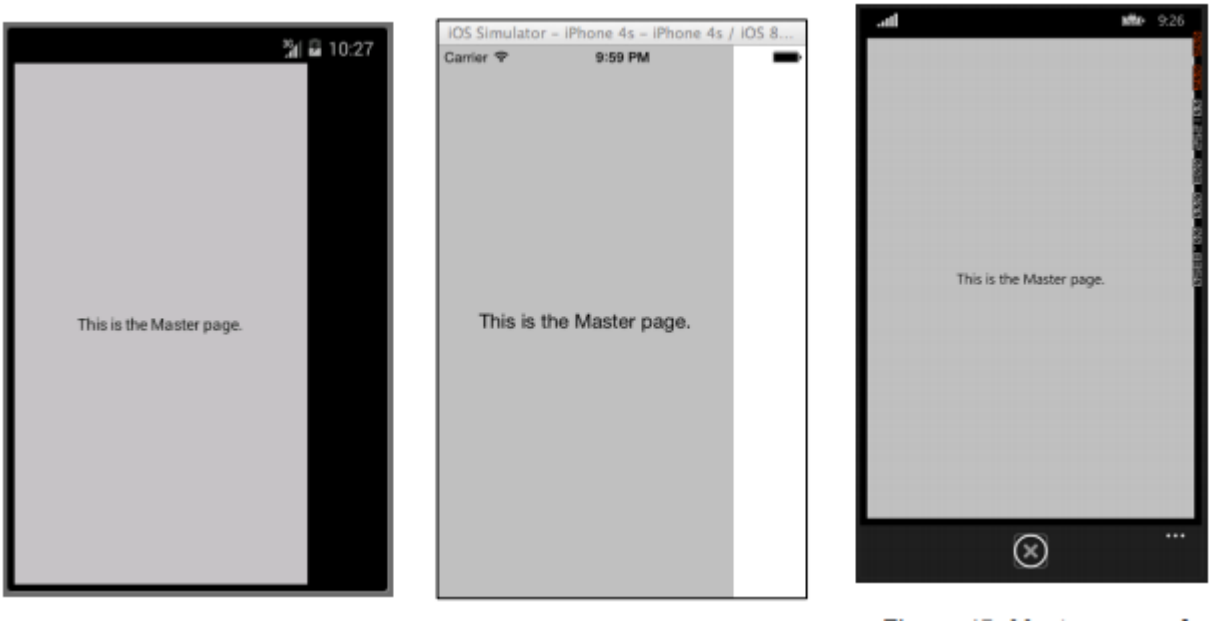
XAML

```
<?xml version="1.0" encoding="utf-8" ?>
<MasterDetailPage xmlns="http://xamarin.com/schemas/2014/forms"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="XamlBasics.SampleXaml">
<MasterDetailPage.Master>
<ContentPage Title = "Master" BackgroundColor = "Silver">
<Label Text="This is the Master page."
TextColor = "Black"
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Master>
<MasterDetailPage.Detail>
<ContentPage>
<Label Text="This is the Detail page."
HorizontalOptions="Center"
VerticalOptions="Center" />
</ContentPage>
</MasterDetailPage.Detail>
</MasterDetailPage>
```

Code

```
var masterDetailPage = new MasterDetailPage {
Master = new ContentPage {
Content = new Label {
Title = "Master",
BackgroundColor = Color.Silver,

TextColor = Color.Black,
Text = "This is the Master page.",
HorizontalOptions = LayoutOptions.Center,
```

```
VerticalOptions = LayoutOptions.Center
}
},
Detail = new ContentPage {
Content = new Label {
Title = "Detail",
Text = "This is the Detail page.",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
}
}
};
```



Xamarin.Forms Seite online lesen: https://riptutorial.com/de/xamarin-forms/topic/7018/xamarin-forms-seite

Kapitel 39: Xamarin.Forms Views

Examples

Taste

Der **Button** ist wahrscheinlich das häufigste Steuerelement nicht nur in mobilen Anwendungen, sondern in allen Anwendungen, die über eine Benutzeroberfläche verfügen. Das Konzept einer Schaltfläche hat zu viele Zwecke, um sie hier aufzulisten. Im Allgemeinen verwenden Sie jedoch eine Schaltfläche, mit der Benutzer Aktionen oder Vorgänge in Ihrer Anwendung einleiten können. Diese Operation kann alles umfassen, von der einfachen Navigation in Ihrer App bis zum Senden von Daten an einen Webdienst irgendwo im Internet.

XAML

```
<Button
    x:Name="MyButton"
    Text="Click Me!"
    TextColor="Red"
    BorderColor="Blue"
    VerticalOptions="Center"
    HorizontalOptions="Center"
    Clicked="Button_Clicked"/>
```

XAML Code-Behind

```
public void Button_Clicked( object sender, EventArgs args )
{
    MyButton.Text = "I've been clicked!";
}
```

Code

```
var button = new Button( )
{
    Text = "Hello, Forms !",
    VerticalOptions = LayoutOptions.CenterAndExpand,
    HorizontalOptions = LayoutOptions.CenterAndExpand,
    TextColor = Color.Red,
    BorderColor = Color.Blue,
};

button.Clicked += ( sender, args ) =>
{
    var b = (Button) sender;
    b.Text = "I've been clicked!";
};
```

Datumsauswahl

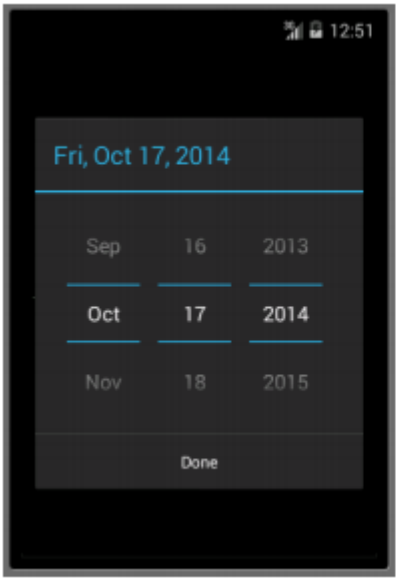
In mobilen Anwendungen wird es häufig Gründe geben, Termine zu behandeln. Wenn Sie mit Datumsangaben arbeiten, benötigen Sie wahrscheinlich eine Art Benutzereingabe, um ein Datum auszuwählen. Dies kann beim Arbeiten mit einer Zeitplan- oder Kalender-App auftreten. In diesem Fall ist es am besten, den Benutzern ein spezielles Steuerelement bereitzustellen, mit dem sie interaktiv ein Datum auswählen können, anstatt manuell ein Datum eingeben zu müssen. Hier ist das DatePicker-Steuerelement wirklich nützlich.

XAML

```
<DatePicker Date="09/12/2014" Format="d" />
```

Code

```
var datePicker = new DatePicker{
Date = DateTime.Now,
Format = "d"
};
```



Eintrag

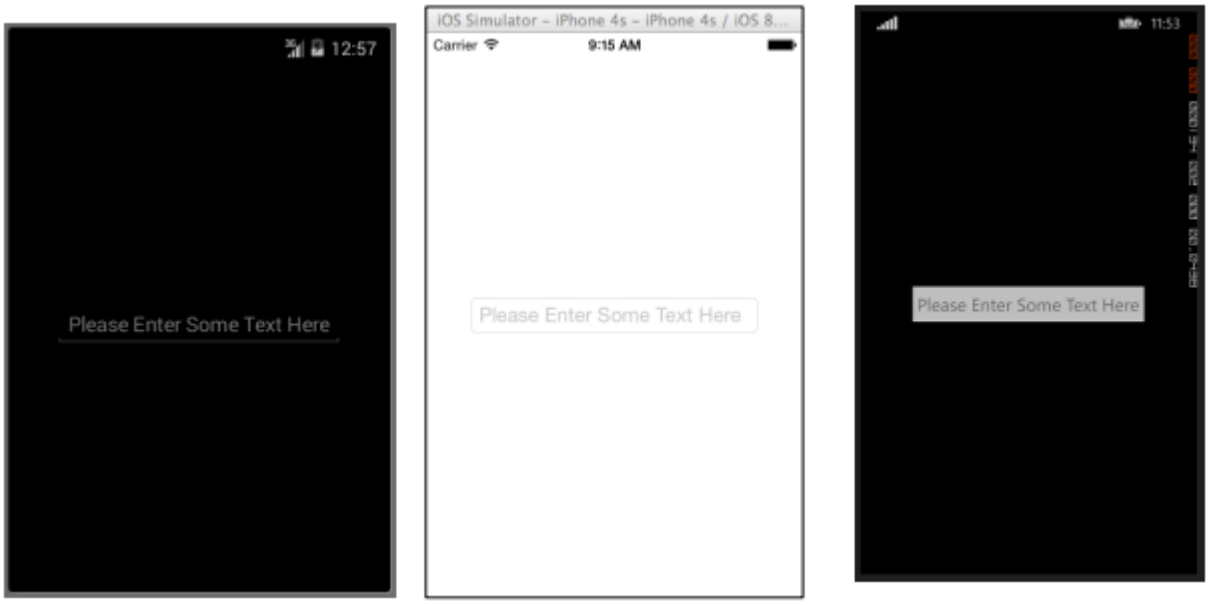
In der Eintragsansicht können Benutzer eine einzelne Textzeile eingeben. Diese einzelne Textzeile kann für mehrere Zwecke verwendet werden, z. B. zur Eingabe von grundlegenden Notizen, Anmeldeinformationen, URLs und mehr. Diese Ansicht ist eine Mehrzweckansicht. Das heißt, wenn Sie normalen Text eingeben oder ein Kennwort verdecken möchten, geschieht dies alles mit diesem einzigen Steuerelement.

XAML

```
<Entry Placeholder="Please Enter Some Text Here"
HorizontalOptions="Center"
VerticalOptions="Center"
Keyboard="Email"/>
```

Code

```
var entry = new Entry {
Placeholder = "Please Enter Some Text Here",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center,
Keyboard = Keyboard.Email
};
```



Editor

Der Editor ist dem Eintrag sehr ähnlich, da er Benutzern die Eingabe von Freiform-Text ermöglicht. Der Unterschied besteht darin, dass der Editor eine mehrzeilige Eingabe zulässt, während der Eintrag nur für die einzeilige Eingabe verwendet wird. Der Eintrag enthält auch einige weitere Eigenschaften als der Editor, um die Ansicht weiter anpassen zu können.

XAML

```
<Editor HorizontalOptions="Fill"
VerticalOptions="Fill"
 Keyboard="Chat"/>
```

Code

```
var editor = new Editor {
HorizontalOptions = LayoutOptions.Fill,
VerticalOptions = LayoutOptions.Fill,
Keyboard = Keyboard.Chat
};
```

Bild

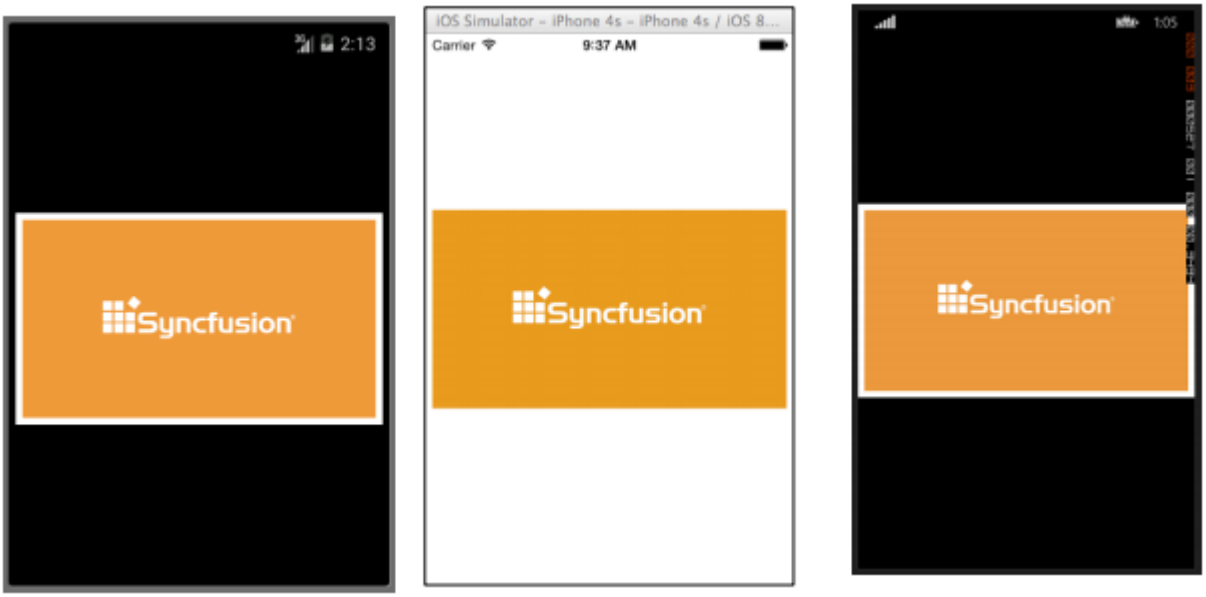
Bilder sind sehr wichtige Bestandteile jeder Anwendung. Sie bieten die Möglichkeit, zusätzliche visuelle Elemente sowie Branding in Ihre Anwendung einzufügen. Ganz zu schweigen davon, dass Bilder in der Regel interessanter sind als Text oder Schaltflächen. Sie können ein Image als eigenständiges Element in Ihrer Anwendung verwenden, ein Image-Element kann jedoch auch zu anderen View-Elementen, z. B. einer Schaltfläche, hinzugefügt werden.

XAML

```
<Image Aspect="AspectFit" Source="http://d2g29cya9iq7ip.cloudfront.net/co
ntent/images/company/aboutus-video-bg.png?v=25072014072745"/>
```

Code

```
var image = new Image {
Aspect = Aspect.AspectFit,
Source = ImageSource.FromUri(new Uri("http://d2g29cya9iq7ip.cloudfron
t.net/content/images/company/aboutus-video-bg.png?v=25072014072745"))
};
```



Etikette

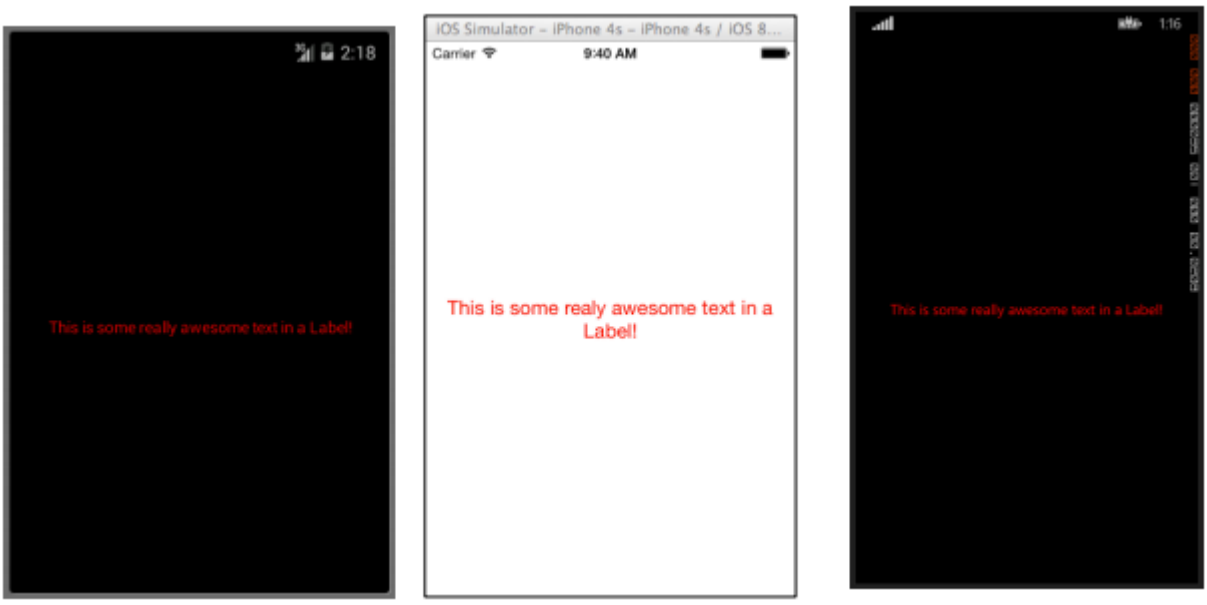
Ob Sie es glauben oder nicht, das Label ist nicht nur in Xamarin.Forms, sondern in der UI-Entwicklung im Allgemeinen eine der wichtigsten, aber unterbewerteten Ansichtsklassen. Es wird als eine ziemlich langweilige Textzeile gesehen, aber ohne diese Textzeile wäre es sehr schwierig, dem Benutzer bestimmte Ideen zu vermitteln. Beschriftungssteuerelemente können verwendet werden, um zu beschreiben, was der Benutzer in ein Editor- oder Eingabesteuerelement eingeben soll. Sie können einen Abschnitt der Benutzeroberfläche beschreiben und ihm den Kontext geben. Sie können verwendet werden, um die Summe in einer Taschenrechner-App anzuzeigen. Ja, das Label ist wirklich das vielseitigste Steuerelement in Ihrer Werkzeugtasche, das nicht immer viel Aufmerksamkeit erregt, aber es ist das erste, das bemerkt wird, wenn es nicht vorhanden ist.

XAML

```
<Label Text="This is some really awesome text in a Label!"
TextColor="Red"
XAlign="Center"
YAlign="Center"/>
```

Code

```
var label = new Label {
Text = "This is some really awesome text in a Label!",
TextColor = Color.Red,
XAlign = TextAlignment.Center,
YAlign = TextAlignment.Center
};
```



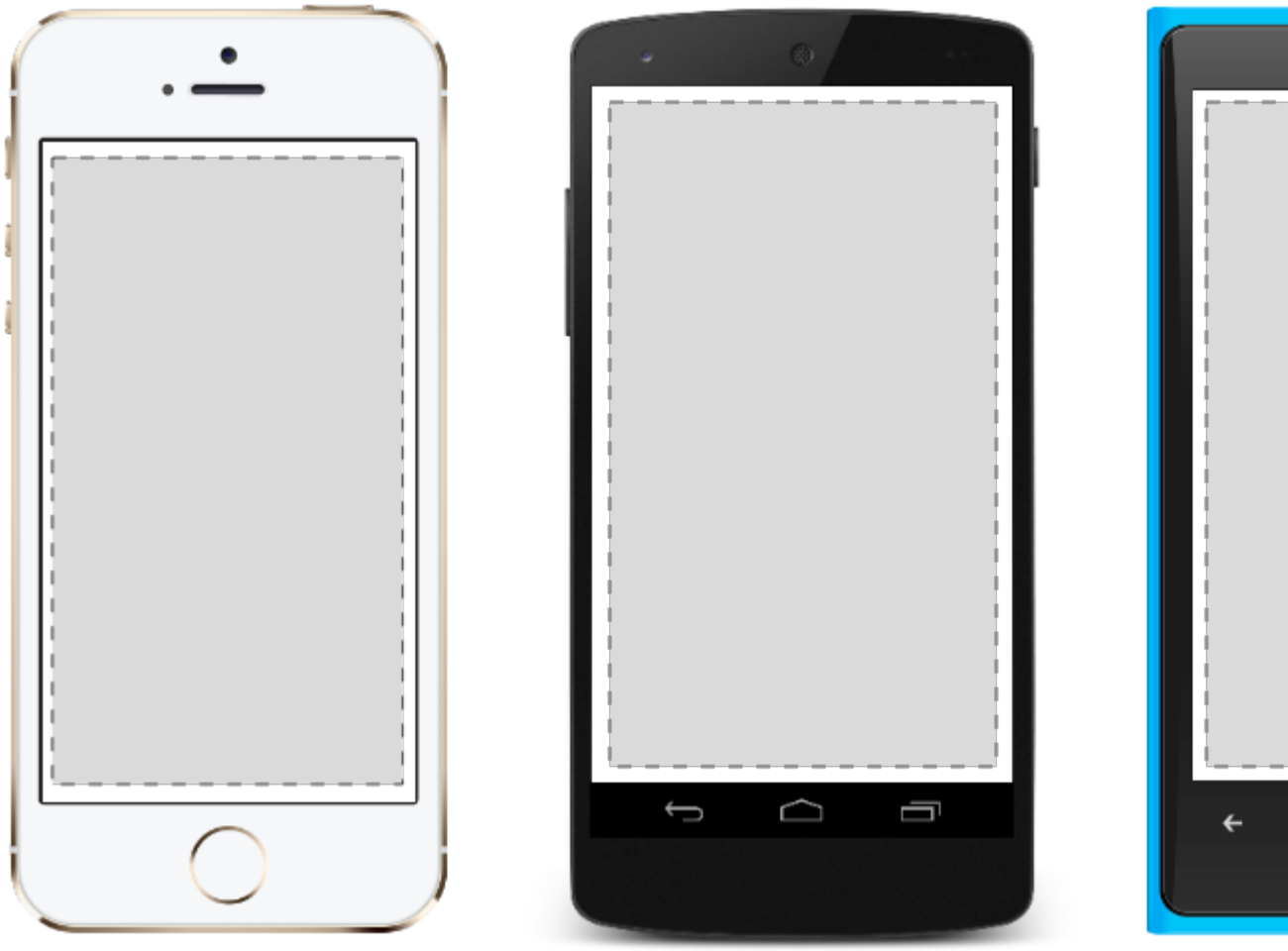
Xamarin.Forms Views online lesen: https://riptutorial.com/de/xamarin-forms/topic/7369/xamarin-forms-views

Kapitel 40: Xamarin-Formularlayouts

Examples

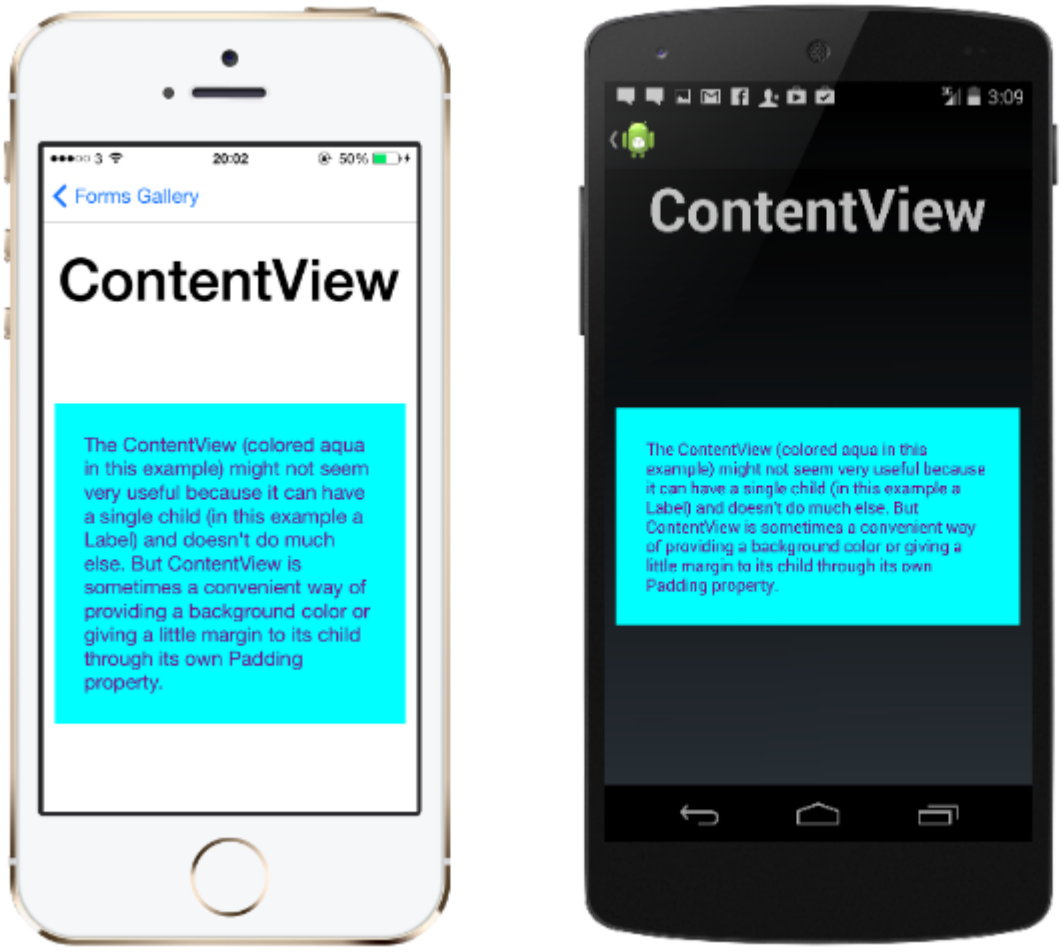
ContentPresenter

Ein Layout-Manager für Vorlagen mit Vorlagen. Wird in einer ControlTemplate verwendet, um zu markieren, wo der anzuzeigende Inhalt angezeigt wird.



ContentView

Ein Element mit einem einzigen Inhalt. ContentView hat wenig Eigennutz. Sein Zweck ist es, als Basisklasse für benutzerdefinierte zusammengesetzte Ansichten zu dienen.



XAML

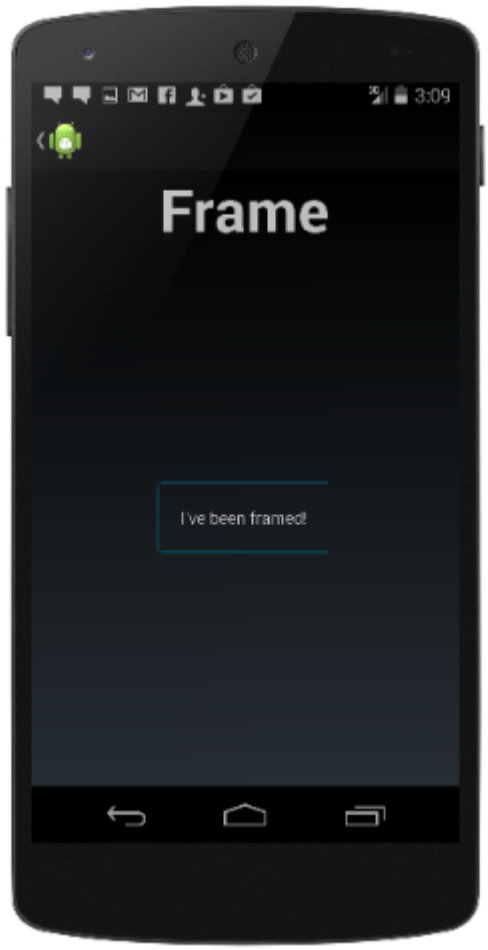
```
<ContentView>
<Label Text="Hi, I'm a simple Label inside of a simple ContentView"
HorizontalOptions="Center"
VerticalOptions="Center"/>
</ContentView>
```

Code

```
var contentView = new ContentView {
Content = new Label {
 Text = "Hi, I'm a simple Label inside of a simple ContentView",
 HorizontalOptions = LayoutOptions.Center,
 VerticalOptions = LayoutOptions.Center
}
};
```

Rahmen

Ein Element, das ein einzelnes Kind enthält, mit einigen Gestaltungsoptionen. Der Rahmen hat einen Standardwert von Xamarin.Forms.Layout.Padding von 20.



XAML

```
<Frame>
<Label Text="I've been framed!"
HorizontalOptions="Center"
VerticalOptions="Center" />
</Frame>
```

Code

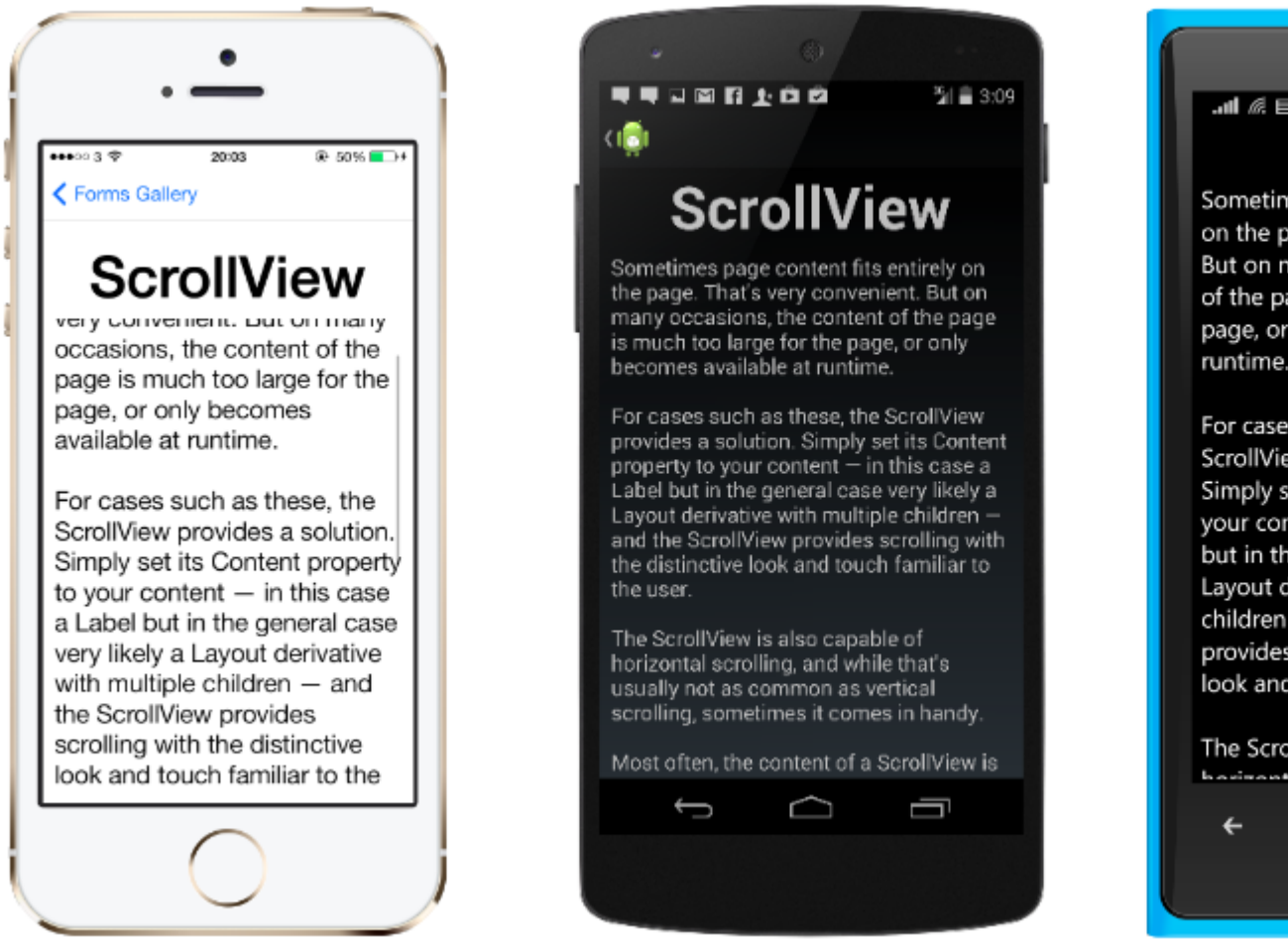
```
var frameView = new Frame {
Content = new Label {
 Text = "I've been framed!",
HorizontalOptions = LayoutOptions.Center,
VerticalOptions = LayoutOptions.Center
},
OutlineColor = Color.Red
};
```

ScrollView

Ein Element, das scrollen kann, wenn der `Content` erfordert.

`ScrollView` enthält Layouts und ermöglicht das `ScrollView` . `ScrollView` wird auch verwendet, um zuzulassen, dass Ansichten automatisch zum sichtbaren Teil des Bildschirms wechseln, wenn die

Tastatur angezeigt wird.



Hinweis: `ScrollViews` sollten nicht verschachtelt sein. Außerdem sollte `ScrollViews` nicht mit anderen Steuerelementen verschachtelt werden, die einen Bildlauf ermöglichen, wie `WebView` . B. `ListView` und `WebView` .

Ein `ScrollView` ist leicht zu definieren. In XAML:

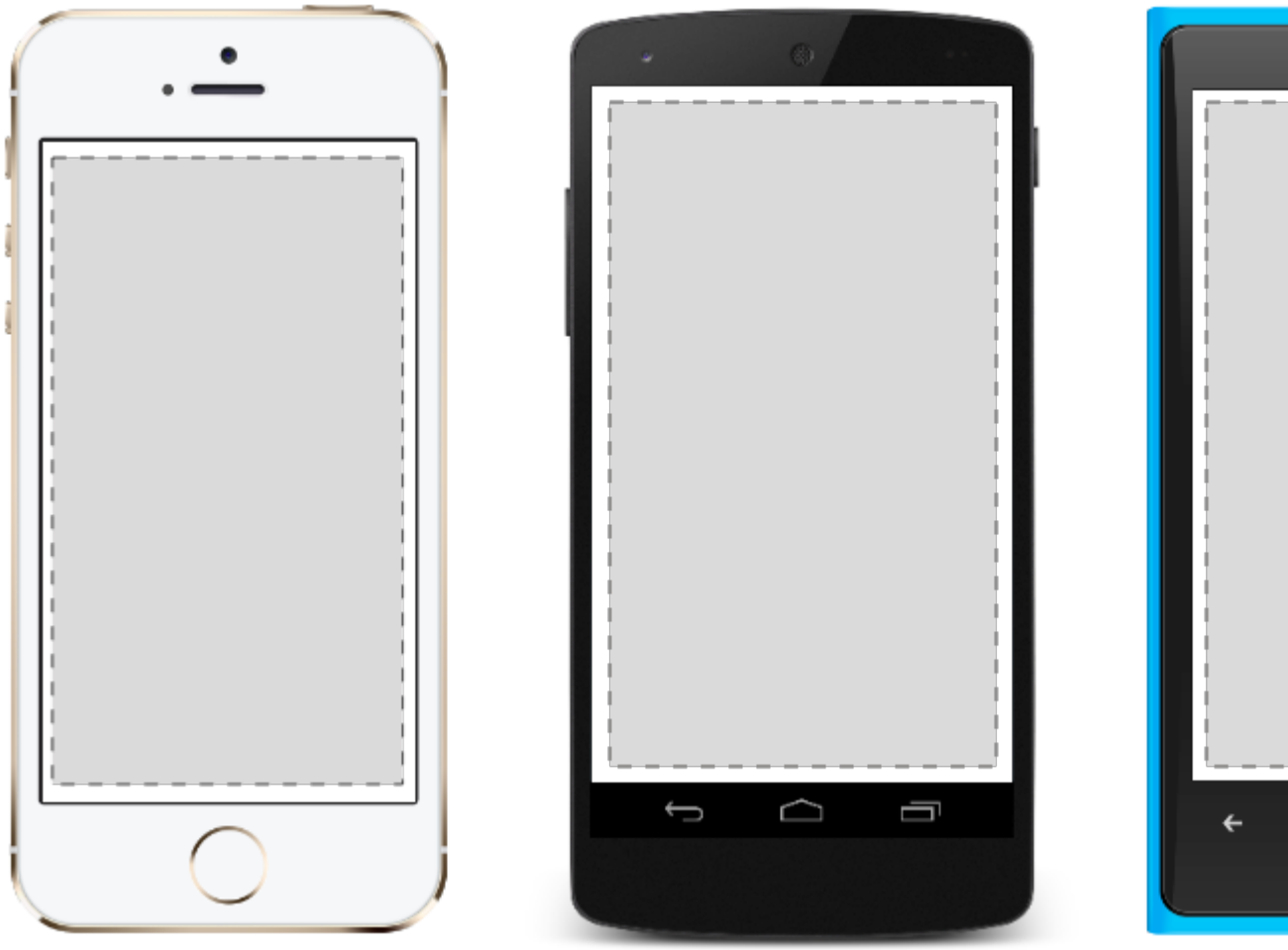
```
<ContentPage.Content>
    <ScrollView>
        <StackLayout>
            <BoxView BackgroundColor="Red" HeightRequest="600" WidthRequest="150" />
            <Entry />
        </StackLayout>
    </ScrollView>
</ContentPage.Content>
```

Dieselbe Definition im Code:

```
var scroll = new ScrollView();
Content = scroll;
var stack = new StackLayout();
stack.Children.Add(new BoxView { BackgroundColor = Color.Red,   HeightRequest = 600,
WidthRequest = 600 });
stack.Children.Add(new Entry());
```

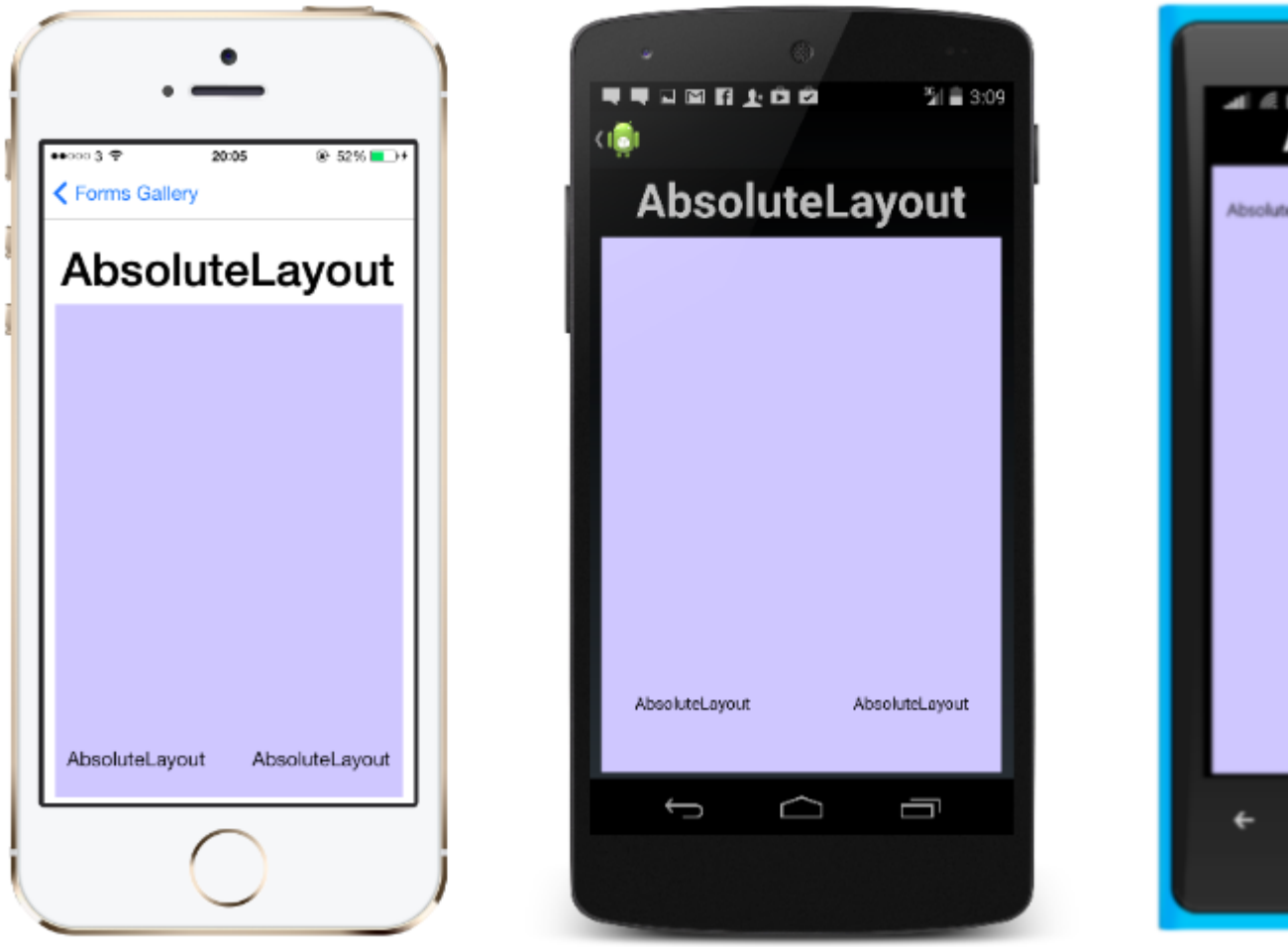
TemplatedView

Ein Element, das Inhalt mit einer Steuerelementvorlage und die Basisklasse für `ContentView` .



AbsoluteLayout

`AbsoluteLayout` positioniert und sortiert untergeordnete Elemente proportional zu ihrer eigenen Größe und Position oder zu absoluten Werten. Untergeordnete Ansichten können mit proportionalen Werten oder statischen Werten positioniert und deren Größe angepasst werden. Proportionale und statische Werte können gemischt werden.



Eine Definition eines `AbsoluteLayout` in XAML sieht folgendermaßen aus:

```
<AbsoluteLayout>
    <Label Text="I'm centered on iPhone 4 but no other device"
        AbsoluteLayout.LayoutBounds="115,150,100,100" LineBreakMode="WordWrap"  />
    <Label Text="I'm bottom center on every device."
        AbsoluteLayout.LayoutBounds=".5,1,.5,.1" AbsoluteLayout.LayoutFlags="All"
        LineBreakMode="WordWrap"  />
    <BoxView Color="Olive"  AbsoluteLayout.LayoutBounds="1,.5, 25, 100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Red" AbsoluteLayout.LayoutBounds="0,.5,25,100"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,100,25"
        AbsoluteLayout.LayoutFlags="PositionProportional" />
    <BoxView Color="Blue" AbsoluteLayout.LayoutBounds=".5,0,1,25"
        AbsoluteLayout.LayoutFlags="PositionProportional, WidthProportional" />
</AbsoluteLayout>
```

Das gleiche Layout würde im Code so aussehen:

```
Title = "Absolute Layout Exploration - Code";
var layout = new AbsoluteLayout();

var centerLabel = new Label {
Text = "I'm centered on iPhone 4 but no other device.",
LineBreakMode = LineBreakMode.WordWrap};
```

```
AbsoluteLayout.SetLayoutBounds (centerLabel, new Rectangle (115, 159, 100, 100));
// No need to set layout flags, absolute positioning is the default

var bottomLabel = new Label { Text = "I'm bottom center on every device.", LineBreakMode =
LineBreakMode.WordWrap };
AbsoluteLayout.SetLayoutBounds (bottomLabel, new Rectangle (.5, 1, .5, .1));
AbsoluteLayout.SetLayoutFlags (bottomLabel, AbsoluteLayoutFlags.All);

var rightBox = new BoxView{ Color = Color.Olive };
AbsoluteLayout.SetLayoutBounds (rightBox, new Rectangle (1, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (rightBox, AbsoluteLayoutFlags.PositionProportional);

var leftBox = new BoxView{ Color = Color.Red };
AbsoluteLayout.SetLayoutBounds (leftBox, new Rectangle (0, .5, 25, 100));
AbsoluteLayout.SetLayoutFlags (leftBox, AbsoluteLayoutFlags.PositionProportional);

var topBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 100, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional);

var twoFlagsBox = new BoxView{ Color = Color.Blue };
AbsoluteLayout.SetLayoutBounds (topBox, new Rectangle (.5, 0, 1, 25));
AbsoluteLayout.SetLayoutFlags (topBox, AbsoluteLayoutFlags.PositionProportional |
AbsoluteLayout.WidthProportional);

layout.Children.Add (bottomLabel);
layout.Children.Add (centerLabel);
layout.Children.Add (rightBox);
layout.Children.Add (leftBox);
layout.Children.Add (topBox);
```

Mit dem Steuerelement AbsoluteLayout in Xamarin.Forms können Sie angeben, wo genau die untergeordneten Elemente auf dem Bildschirm angezeigt werden sollen, sowie deren Größe und Form (Begrenzungen).

Es gibt verschiedene Möglichkeiten, die Grenzen der untergeordneten Elemente basierend auf der AbsoluteLayoutFlags-Enumeration festzulegen, die während dieses Prozesses verwendet werden. Die Aufzählung **AbsoluteLayoutFlags** enthält die folgenden Werte:

- **Alle** : Alle Maße sind proportional.
- **HeightProportional** : Die Höhe ist proportional zum Layout.
- **Keine** : Es wird keine Interpretation vorgenommen.
- **PositionProportional** : Kombiniert XProportional und YProportional.
- **SizeProportional** : Kombiniert WidthProportional und HeightProportional.
- **WidthProportional** : Die Breite ist proportional zum Layout.
- **XProportional** : X-Eigenschaft ist proportional zum Layout.
- **YProportional** : Die Y-Eigenschaft ist proportional zum Layout.

Die Arbeit mit dem Layout des AbsoluteLayout-Containers mag auf den ersten Blick etwas uninteressant erscheinen, aber mit ein wenig Einsatz wird er vertraut. Nachdem Sie Ihre untergeordneten Elemente erstellt haben, müssen Sie drei Schritte ausführen, um sie an eine absolute Position innerhalb des Containers zu setzen. Sie möchten die den Elementen zugewiesenen Flags mit der **AbsoluteLayout.SetLayoutFlags ()** -Methode festlegen. Sie möchten auch die **AbsoluteLayout.SetLayoutBounds () -** Methode verwenden, um den

Elementen ihre Grenzen zu geben. Schließlich möchten Sie die untergeordneten Elemente der Children-Sammlung hinzufügen. Da Xamarin.Forms eine Abstraktionsschicht zwischen Xamarin und den gerätespezifischen Implementierungen ist, können die Positionswerte unabhängig von den Gerätepixeln sein. Hier kommen die zuvor genannten Layout-Flags ins Spiel. Sie können auswählen, wie der Layoutprozess der Xamarin.Forms-Steuerelemente die von Ihnen definierten Werte interpretieren soll.

Gitter

Ein Layout mit Ansichten, die in Zeilen und Spalten angeordnet sind.

Dies ist eine typische `Grid` Definition in XAML.

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition Height="*" />
    <RowDefinition Height="200" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>

  <ContentView Grid.Row="0" Grid.Column="0"/>
```

```
  <ContentView Grid.Row="1" Grid.Column="0"/>
  <ContentView Grid.Row="2" Grid.Column="0"/>

  <ContentView Grid.Row="0" Grid.Column="1"/>
  <ContentView Grid.Row="1" Grid.Column="1"/>
  <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

Das gleiche im Code definierte `Grid` sieht folgendermaßen aus:

```
var grid = new Grid();
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(2, GridUnitType.Star) });
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength (1, GridUnitType.Star)
});
grid.RowDefinitions.Add (new RowDefinition { Height = new GridLength(200)});
grid.ColumnDefinitions.Add (new ColumnDefinition{ Width = new GridLength (200) });
```

So fügen Sie Elemente zum Raster hinzu: In `XAML` :

```
<Grid>

   <--DEFINITIONS...--!>

   <ContentView Grid.Row="0" Grid.Column="0"/>
   <ContentView Grid.Row="1" Grid.Column="0"/>
   <ContentView Grid.Row="2" Grid.Column="0"/>

   <ContentView Grid.Row="0" Grid.Column="1"/>
   <ContentView Grid.Row="1" Grid.Column="1"/>
   <ContentView Grid.Row="2" Grid.Column="1"/>

</Grid>
```

In C # -Code:

```
var grid = new Grid();
//DEFINITIONS...
var topLeft = new Label { Text = "Top Left" };
var topRight = new Label { Text = "Top Right" };
var bottomLeft = new Label { Text = "Bottom Left" };
var bottomRight = new Label { Text = "Bottom Right" };
grid.Children.Add(topLeft, 0, 0);
grid.Children.Add(topRight, 0, 1);
grid.Children.Add(bottomLeft, 1, 0);
grid.Children.Add(bottomRight, 1, 1);
```

Für `Height` und `Width` mehrere Einheiten zur Verfügung.

- **Auto** - Größen automatisch Inhalt in der Zeile oder Spalte zu passen. Wird in C # als GridUnitType.Auto oder in XAML als Auto angegeben.
- **Proportional** - Größe der Spalten und Zeilen im Verhältnis zum verbleibenden Platz. Wird als Wert und GridUnitType.Star in C # und als # * in XAML angegeben, wobei # der gewünschte Wert ist. Wenn Sie eine Zeile / Spalte mit * angeben, füllt sie den verfügbaren Platz.

- **Absolut** - Größe von Spalten und Zeilen mit bestimmten festen Werten für Höhe und Breite. Wird als Wert und als GridUnitType.Absolute in C # und als # in XAML angegeben, wobei # der gewünschte Wert ist.

Hinweis: Die Breitenwerte für Spalten werden standardmäßig in Xamarin.Forms auf Auto festgelegt. Dies bedeutet, dass die Breite von der Größe der untergeordneten Elemente bestimmt wird. Beachten Sie, dass dies von der Implementierung von XAML auf Microsoft-Plattformen abweicht, bei denen die Standardbreite * ist, wodurch der verfügbare Speicherplatz gefüllt wird.

RelativesLayout

Ein `Layout` , das `Constraints` für das Layout der `Constraints Layout` verwendet.

`RelativeLayout` wird verwendet, um Ansichten relativ zu Eigenschaften des Layouts oder der gleichgeordneten Ansichten zu positionieren und deren Größe zu ändern. Im Gegensatz zu `AbsoluteLayout` verfügt `RelativeLayout` nicht über das Konzept des beweglichen Ankers und hat keine Möglichkeit, Elemente relativ zum unteren oder rechten Rand des Layouts zu positionieren. `RelativeLayout` unterstützt die Positionierung von Elementen außerhalb der eigenen Grenzen.

Ein relativesLayout in XAML sieht folgendermaßen aus:

```
<RelativeLayout>
    <BoxView Color="Red" x:Name="redBox"
```

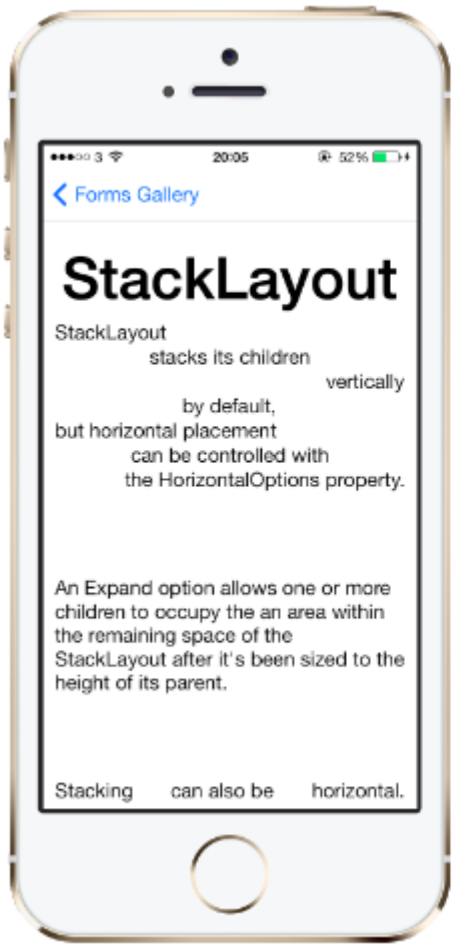
```
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToParent,
            Property=Height,Factor=.15,Constant=0}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=1,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.8,Constant=0}" />
    <BoxView Color="Blue"
        RelativeLayout.YConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=Y,Factor=1,Constant=20}"
        RelativeLayout.XConstraint="{ConstraintExpression Type=RelativeToView,
            ElementName=redBox,Property=X,Factor=1,Constant=20}"
        RelativeLayout.WidthConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Width,Factor=.5,Constant=0}"
        RelativeLayout.HeightConstraint="{ConstraintExpression
            Type=RelativeToParent,Property=Height,Factor=.5,Constant=0}" />
</RelativeLayout>
```

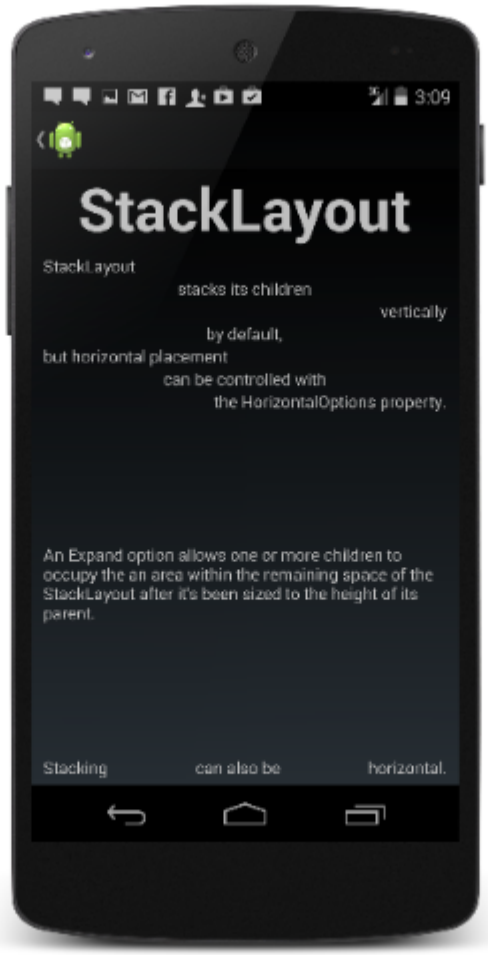
Dasselbe Layout kann mit diesem Code erreicht werden:

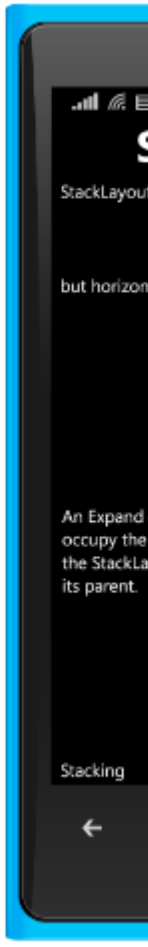
```
layout.Children.Add (redBox, Constraint.RelativeToParent ((parent) => {
        return parent.X;
    }), Constraint.RelativeToParent ((parent) => {
        return parent.Y * .15;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .8;
    }));
layout.Children.Add (blueBox, Constraint.RelativeToView (redBox, (Parent, sibling) => {
        return sibling.X + 20;
    }), Constraint.RelativeToView (blueBox, (parent, sibling) => {
        return sibling.Y + 20;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Width * .5;
    }), Constraint.RelativeToParent((parent) => {
        return parent.Height * .5;
    }));
```

StackLayout

`StackLayout` organisiert Ansichten in einer eindimensionalen Linie ("Stack") entweder horizontal oder vertikal. `Views` in einem `StackLayout` können basierend auf dem Platz im Layout unter Verwendung der `StackLayout` in der Größe `StackLayout` werden. Die Positionierung wird dadurch bestimmt, dass die Bestellansichten zum Layout und zu den Layoutoptionen der Ansichten hinzugefügt wurden.







Verwendung in XAML

```
<StackLayout>
    <Label Text="This will be on top" />
    <Button Text="This will be on the bottom" />
</StackLayout>
```

Verwendung im Code

```
StackLayout stackLayout = new StackLayout
{
    Spacing = 0,
    VerticalOptions = LayoutOptions.FillAndExpand,
    Children =
    {
        new Label
        {
            Text = "StackLayout",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
            Text = "stacks its children",
```

```
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "vertically",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "by default,",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "but horizontal placement",
            HorizontalOptions = LayoutOptions.Start
        },
        new Label
        {
            Text = "can be controlled with",
            HorizontalOptions = LayoutOptions.Center
        },
        new Label
        {
            Text = "the HorizontalOptions property.",
            HorizontalOptions = LayoutOptions.End
        },
        new Label
        {
            Text = "An Expand option allows one or more children " +
                   "to occupy the an area within the remaining " +
                   "space of the StackLayout after it's been sized " +
                   "to the height of its parent.",
            VerticalOptions = LayoutOptions.CenterAndExpand,
            HorizontalOptions = LayoutOptions.End
        },
        new StackLayout
        {
            Spacing = 0,
            Orientation = StackOrientation.Horizontal,
            Children =
            {
                new Label
                {
                    Text = "Stacking",
                },
                new Label
                {
                    Text = "can also be",
                    HorizontalOptions = LayoutOptions.CenterAndExpand
                },
                new Label
                {
                    Text = "horizontal.",
                },
            }
        }
    }
};
```

Xamarin-Formularlayouts online lesen: https://riptutorial.com/de/xamarin-

forms/topic/6273/xamarin-formularlayouts

Kapitel 41: Xamarin-Geste

Examples

Tippen Sie auf Geste

Mit der Tap-Geste können Sie jedes UI-Element anklickbar machen (Bilder, Buttons, StackLayouts, ...):

(1) Im Code unter Verwendung des Ereignisses:

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.Tapped += (s, e) => {
    // handle the tap
};
image.GestureRecognizers.Add(tapGestureRecognizer);
```

(2) Im Code mit `ICommand` (zum Beispiel mit MVVM-Pattern):

```
var tapGestureRecognizer = new TapGestureRecognizer();
tapGestureRecognizer.SetBinding (TapGestureRecognizer.CommandProperty, "TapCommand");
image.GestureRecognizers.Add(tapGestureRecognizer);
```

(3) Oder in Xaml (mit event und `ICommand` wird nur einer benötigt):

```
<Image Source="tapped.jpg">
    <Image.GestureRecognizers>
        <TapGestureRecognizer Tapped="OnTapGestureRecognizerTapped" Command="{Binding
TapCommand"} />
  </Image.GestureRecognizers>
</Image>
```

Xamarin-Geste online lesen: https://riptutorial.com/de/xamarin-forms/topic/7994/xamarin-geste

Kapitel 42: Xamarin-Geste

Examples

Gestenereignis

Wenn wir die Kontrolle über Label übernehmen, bietet das Label kein Ereignis. <Label x: Name = "lblSignUp Text =" Sie haben noch kein Konto? "/>.

Wenn der Benutzer Button durch Label ersetzen möchte, geben wir das Ereignis für Label an. Wie nachfolgend dargestellt:

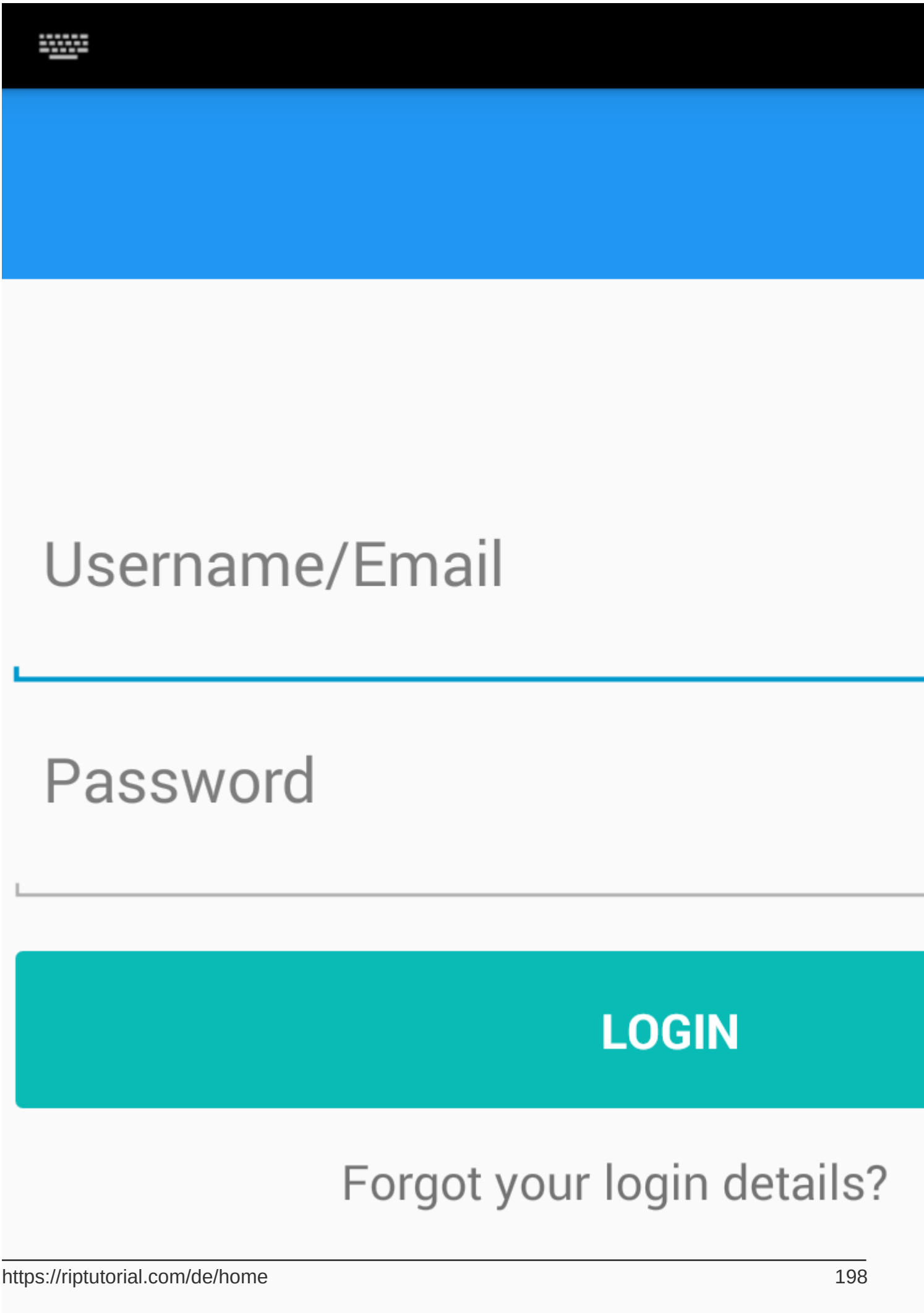
XAML

```
<Label x:Name="lblSignUp" Text="Don't have an account?" Grid.Row="8" Grid.Column="1"
Grid.ColumnSpan="2">
  <Label.GestureRecognizers>
    <TapGestureRecognizer
   Tapped="lblSignUp_Tapped"/>
  </Label.GestureRecognizers>
```

C #

```
var lblSignUp_Tapped = new TapGestureRecognizer();
lblSignUp_Tapped.Tapped += (s,e) =>
{
//
//  Do your work here.
//
};
lblSignUp.GestureRecognizers.Add(lblSignUp_Tapped);
```

Der Bildschirm unten zeigt das Label-Ereignis. Bildschirm 1: Das Label "Sie haben noch kein Konto?" wie unten gezeigt.


Username/Email
Password
LOGIN
Forgot your login details?

Weitere Informationen: [https://developer.xamarin.com/guides/xamarin-forms/user-interface/gestures/tap/([1]

Xamarin-Geste online lesen: https://riptutorial.com/de/xamarin-forms/topic/8009/xamarin-geste

Kapitel 43: Zugriff auf native Funktionen mit DependencyService

Bemerkungen

Wenn der Code nicht beschädigt werden soll, wenn keine Implementierung gefunden wird, überprüfen Sie zuerst den `DependencyService` ob eine Implementierung verfügbar ist.

Sie können dies durch eine einfache Überprüfung tun, wenn es nicht `null` .

```
var speaker = DependencyService.Get<ITextToSpeech>();

if (speaker != null)
{
    speaker.Speak("Ready for action!");
}
```

oder, wenn Ihre IDE C # 6 unterstützt, mit einem nullbedingten Operator:

```
var speaker = DependencyService.Get<ITextToSpeech>();

speaker?.Speak("Ready for action!");
```

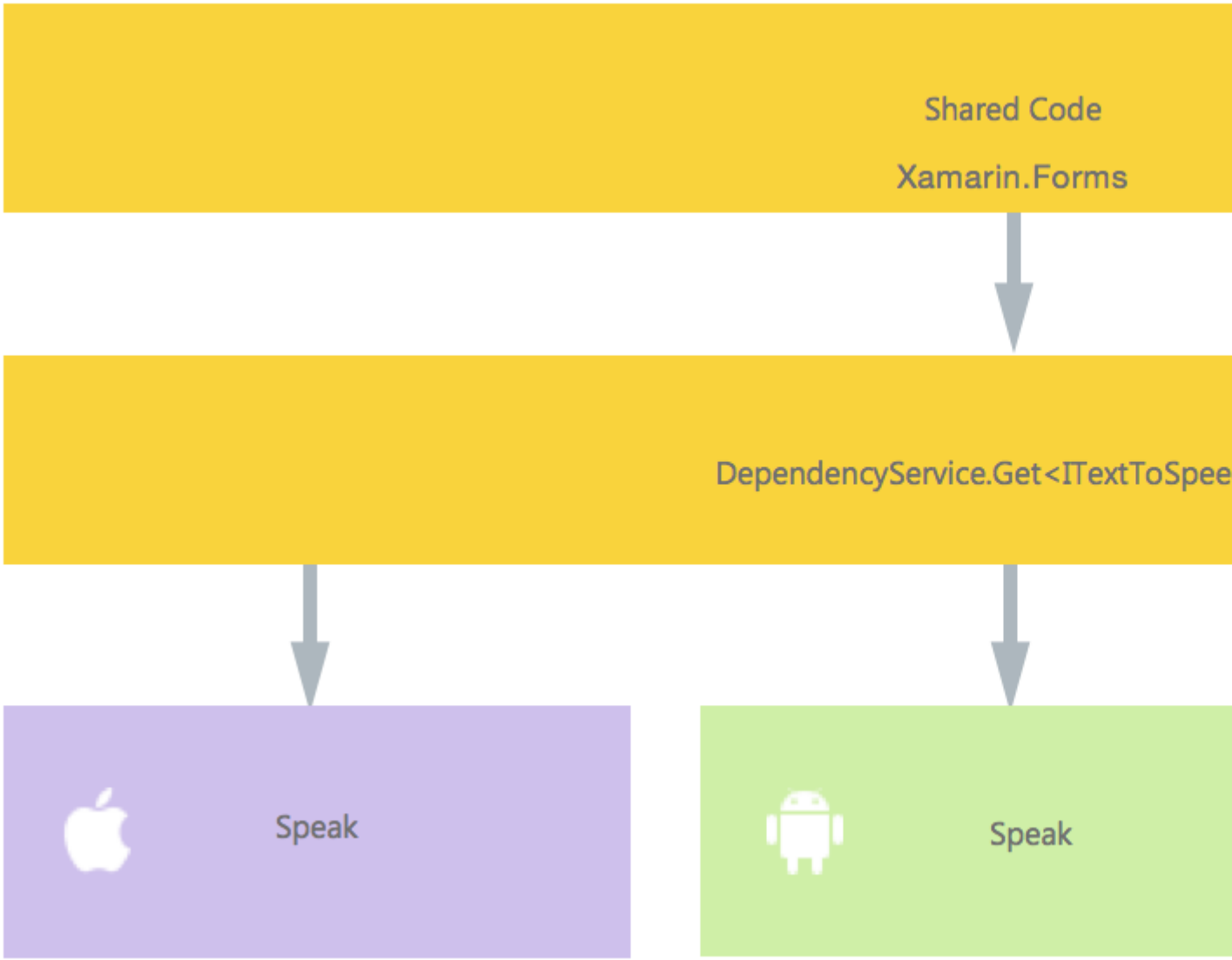
Wenn Sie dies nicht tun und zur Laufzeit keine Implementierung gefunden wird, generiert Ihr Code eine Ausnahme.

Examples

Implementierung von Text-zu-Sprache

Ein gutes Beispiel für eine Funktion, die plattformspezifischen Code anfordert, ist die Implementierung von Text-to-Speech (tts). In diesem Beispiel wird davon ausgegangen, dass Sie mit gemeinsamem Code in einer PCL-Bibliothek arbeiten.

Eine schematische Übersicht unserer Lösung würde wie das Bild darunter aussehen.



In unserem Shared Code definieren wir eine Schnittstelle, die beim `DependencyService` registriert ist. Hier werden wir unsere Forderungen erledigen. Definieren Sie eine Schnittstelle wie darunter.

```
public interface ITextToSpeech
{
    void Speak (string text);
}
```

Jetzt müssen wir in jeder spezifischen Plattform eine Implementierung dieser Schnittstelle erstellen. Beginnen wir mit der iOS-Implementierung.

iOS-Implementierung

```
using AVFoundation;

public class TextToSpeechImplementation : ITextToSpeech
{
```

```
    public TextToSpeechImplementation () {}

    public void Speak (string text)
    {
        var speechSynthesizer = new AVSpeechSynthesizer ();

        var speechUtterance = new AVSpeechUtterance (text) {
            Rate = AVSpeechUtterance.MaximumSpeechRate/4,
            Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
            Volume = 0.5f,
            PitchMultiplier = 1.0f
        };

        speechSynthesizer.SpeakUtterance (speechUtterance);
    }
}
```

Im obigen Codebeispiel stellen Sie fest, dass für iOS spezifischer Code vorhanden ist. `AVSpeechSynthesizer` Typen wie `AVSpeechSynthesizer` . Diese funktionieren nicht im gemeinsam genutzten Code.

Um diese Implementierung beim Xamarin `DependencyService` zu registrieren, fügen Sie dieses Attribut oberhalb der Namespace-Deklaration ein.

```
using AVFoundation;
using DependencyServiceSample.iOS;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.iOS {
    public class TextToSpeechImplementation : ITextToSpeech
//... Rest of code
```

Wenn Sie nun in Ihrem gemeinsam genutzten Code einen solchen Aufruf durchführen, wird die richtige Implementierung für die Plattform, auf der Sie Ihre App ausführen, eingefügt.

`DependencyService.Get<ITextToSpeech>()` . Mehr dazu später.

Android-Implementierung

Die Android-Implementierung dieses Codes würde darunter aussehen.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;

public class TextToSpeechImplementation : Java.Lang.Object, ITextToSpeech,
TextToSpeech.IOnInitListener
{
    TextToSpeech speaker;
    string toSpeak;

    public TextToSpeechImplementation () {}
```

```
    public void Speak (string text)
    {
        var ctx = Forms.Context; // useful for many Android SDK features
        toSpeak = text;
        if (speaker == null) {
            speaker = new TextToSpeech (ctx, this);
        } else {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }

    #region IOnInitListener implementation
    public void OnInit (OperationResult status)
    {
        if (status.Equals (OperationResult.Success)) {
            var p = new Dictionary<string,string> ();
            speaker.Speak (toSpeak, QueueMode.Flush, p);
        }
    }
    #endregion
}
```

Vergessen Sie auch nicht, es beim `DependencyService` zu registrieren.

```
using Android.Speech.Tts;
using Xamarin.Forms;
using System.Collections.Generic;
using DependencyServiceSample.Droid;


[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.Droid{
    //... Rest of code
```

Windows Phone-Implementierung

Schließlich kann dieser Code für Windows Phone verwendet werden.

```
public class TextToSpeechImplementation : ITextToSpeech
{
    public TextToSpeechImplementation() {}

    public async void Speak(string text)
    {
        MediaElement mediaElement = new MediaElement();

        var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

        SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync("Hello World");

        mediaElement.SetSource(stream, stream.ContentType);
        mediaElement.Play();
        await synth.SynthesizeTextToStreamAsync(text);
    }
}
```

Und vergessen Sie nicht noch einmal, es zu registrieren.

```
using Windows.Media.SpeechSynthesis;
using Windows.UI.Xaml.Controls;
using DependencyServiceSample.WinPhone;//enables registration outside of namespace

[assembly: Xamarin.Forms.Dependency (typeof (TextToSpeechImplementation))]
namespace DependencyServiceSample.WinPhone{
    //... Rest of code
```

Implementierung in Shared Code

Jetzt ist alles vorhanden, damit es funktioniert! Schließlich können Sie diese Funktion in Ihrem Shared Code jetzt über die Schnittstelle aufrufen. Zur Laufzeit wird die Implementierung eingefügt, die der aktuellen Plattform entspricht, auf der sie ausgeführt wird.

In diesem Code sehen Sie eine Seite, die sich in einem Xamarin Forms-Projekt befinden könnte. Es erstellt eine Schaltfläche, die die `Speak()` -Methode mithilfe des `DependencyService` aufruft.

```
public MainPage ()
{
    var speak = new Button {
        Text = "Hello, Forms !",
        VerticalOptions = LayoutOptions.CenterAndExpand,
        HorizontalOptions = LayoutOptions.CenterAndExpand,
    };
    speak.Clicked += (sender, e) => {
        DependencyService.Get<ITextToSpeech>().Speak("Hello from Xamarin Forms");
    };
    Content = speak;
}
```

Das Ergebnis ist, dass beim Ausführen der App und Klicken der Schaltfläche der bereitgestellte Text gesprochen wird.

All dies ohne harte Dinge wie Compiler-Tipps und ähnliches zu tun. Sie haben jetzt eine einheitliche Möglichkeit, auf plattformspezifische Funktionen durch plattformunabhängigen Code zuzugreifen.

Versionsnummern von Anwendungs- und Geräte-Betriebssystemen erhalten - Android & iOS - PCL

Im folgenden Beispiel werden die Betriebssystemversionsnummer des Geräts und die Version der Anwendung (die in den Eigenschaften der einzelnen Projekte definiert ist) **erfasst, die** unter Android unter **Version** und unter iOS unter **Version** eingegeben wird.

Erstellen Sie zunächst eine Schnittstelle in Ihrem PCL-Projekt:

```
public interface INativeHelper {
    /// <summary>
```

```
    /// On iOS, gets the <c>CFBundleVersion</c> number and on Android, gets the
<c>PackageInfo</c>'s <c>VersionName</c>, both of which are specified in their respective
project properties.
    /// </summary>
    /// <returns><c>string</c>, containing the build number.</returns>
    string GetAppVersion();

    /// <summary>
    /// On iOS, gets the <c>UIDevice.CurrentDevice.SystemVersion</c> number and on Android,
gets the <c>Build.VERSION.Release</c>.
    /// </summary>
    /// <returns><c>string</c>, containing the OS version number.</returns>
    string GetOsVersion();
}
```

Jetzt implementieren wir die Schnittstelle in Android- und iOS-Projekten.

Android:

```
[assembly: Dependency(typeof(NativeHelper_Android))]

namespace YourNamespace.Droid{
    public class NativeHelper_Android : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() {
            Context context = Forms.Context;
            return context.PackageManager.GetPackageInfo(context.PackageName, 0).VersionName;
        }

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return Build.VERSION.Release; }
    }
}
```

iOS:

```
[assembly: Dependency(typeof(NativeHelper_iOS))]

namespace YourNamespace.iOS {
    public class NativeHelper_iOS : INativeHelper {

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetAppVersion() { return
Foundation.NSBundle.MainBundle.InfoDictionary[new
Foundation.NSString("CFBundleVersion")].ToString(); }

        /// <summary>
        /// See interface summary.
        /// </summary>
        public string GetOsVersion() { return UIDevice.CurrentDevice.SystemVersion; }
    }
```

```
}
```

Nun, um den Code in einer Methode zu verwenden:

```
public string GetOsAndAppVersion {
    INativeHelper helper = DependencyService.Get<INativeHelper>();

    if(helper != null) {
        string osVersion  = helper.GetOsVersion();
        string appVersion = helper.GetBuildNumber()
    }
}
```

Zugriff auf native Funktionen mit DependencyService online lesen: https://riptutorial.com/de/xamarin-forms/topic/2409/zugriff-auf-native-funktionen-mit-dependencyservice

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Xamarin.Forms	Akshay Kulkarni, chrisntr, Community, Demitrian, hankide, jdstaerk, Manohar, patridge, Sergey Metlov, spaceplane
2	Abhängigkeitsdienste	RIYAZ
3	Alert anzeigen	aboozz pallikara, GvSharma, Sreeraj, Yehor Hromadskyi
4	AppSettings Reader in Xamarin.Forms	Ben Ishiyama-Levy, GvSharma
5	Auslöser und Verhalten	hamalaiv, hvaughan3
6	Ausnahmebehandlung	Yehor Hromadskyi
7	Auswirkungen	Swaminathan Vetri
8	BDD-Unit-Tests in Xamarin.Forms	Ben Ishiyama-Levy
9	Benutzerdefinierte Renderer	Bonelol, hankide, Nicolas Bodin-Ripert, Nicolas Bodin-Ripert, nishantvodoo, Yehor Hromadskyi, Zverev Eugene
10	Benutzerdefinierte Schriftarten in Stilen	Roma Rudyak
11	Benutzerdefinierte Steuerelemente erstellen	hvaughan3, spaceplane, Yehor Hromadskyi
12	Caching	Sergey Metlov
13	CarouselView - Vorabversion	dpserge
14	Datenbindung	Andrew, Matthew, Yehor Hromadskyi
15	DependencyService	Steven Thewissen
16	Generischer Xamarin.Forms-App-Lebenszyklus? Plattformabhängig!	Zverev Eugene

17	Gesten	doerig, Gerald Versluis, Michael Rumpler
18	Kontaktauswahl - Xamarin-Formulare (Android und iOS)	Pucho Eric
19	ListViews verwenden	cvanbeek
20	MessagingCenter	Gerald Versluis
21	Mit Karten arbeiten	Taras Shevchuk
22	Mit lokalen Datenbanken arbeiten	Luis Beltran, Manohar
23	Mitteilungen	Gerald Versluis, user1568891
24	Navigation in Xamarin.Forms	Fernando Arreguín, jimmgarr, Lucas Moura Veloso, Paul, Sergey Metlov, Taras Shevchuk, Willian D. Andrade
25	OAuth2	Eng Soon Cheah
26	Plattformspezifische visuelle Anpassungen	Alois, GalaxiaGuy, Paul
27	Plattformspezifisches Verhalten	Ege Aydın
28	SQL-Datenbank und API in Xamarin-Formularen.	RIYAZ
29	Unit Testing	jerone, Sergey Metlov
30	Warum Xamarin-Formulare und wann werden Xamarin-Formulare verwendet?	Daniel Krzyczkowski, mike
31	Xamarin Plugin	Eng Soon Cheah
32	Xamarin Relatives Layout	Ege Aydın
33	Xamarin.Forms Cells	Eng Soon Cheah
34	Xamarin.Forms Seite	Eng Soon Cheah
35	Xamarin.Forms Views	Aaron Thompson, Eng Soon Cheah
36	Xamarin-	Eng Soon Cheah, Gerald Versluis, Lucas Moura Veloso

	Formularlayouts	
37	Xamarin-Geste	Joehl
38	Zugriff auf native Funktionen mit DependencyService	Gerald Versluis, hankide, hvaughan3, Sergey Metlov