Kostenloses eBook

# LERNEN Xamarin.iOS

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#xamarin.io

S

# Inhaltsverzeichnis

# Über

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

It is an unofficial and free Xamarin.iOS 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.iOS.



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.iOS

## Bemerkungen

Mit Xamarin.iOS können Sie native iOS-Anwendungen mit denselben Steuerelementen für die Benutzeroberfläche erstellen wie in Objective-C und Xcode, jedoch mit der Flexibilität und Eleganz einer modernen Sprache (C #), der Leistungsfähigkeit der .NET Base Class Library (BCL) und zwei erstklassige IDEs - Xamarin Studio und Visual Studio - an Ihren Fingerspitzen.

Weitere Informationen zum Installieren von Xamarin.iOS auf einem Mac- oder Windows-Computer finden Sie in den Kurzanleitungen für die ersten Schritte im Xamarin Developer Center

## Versionen

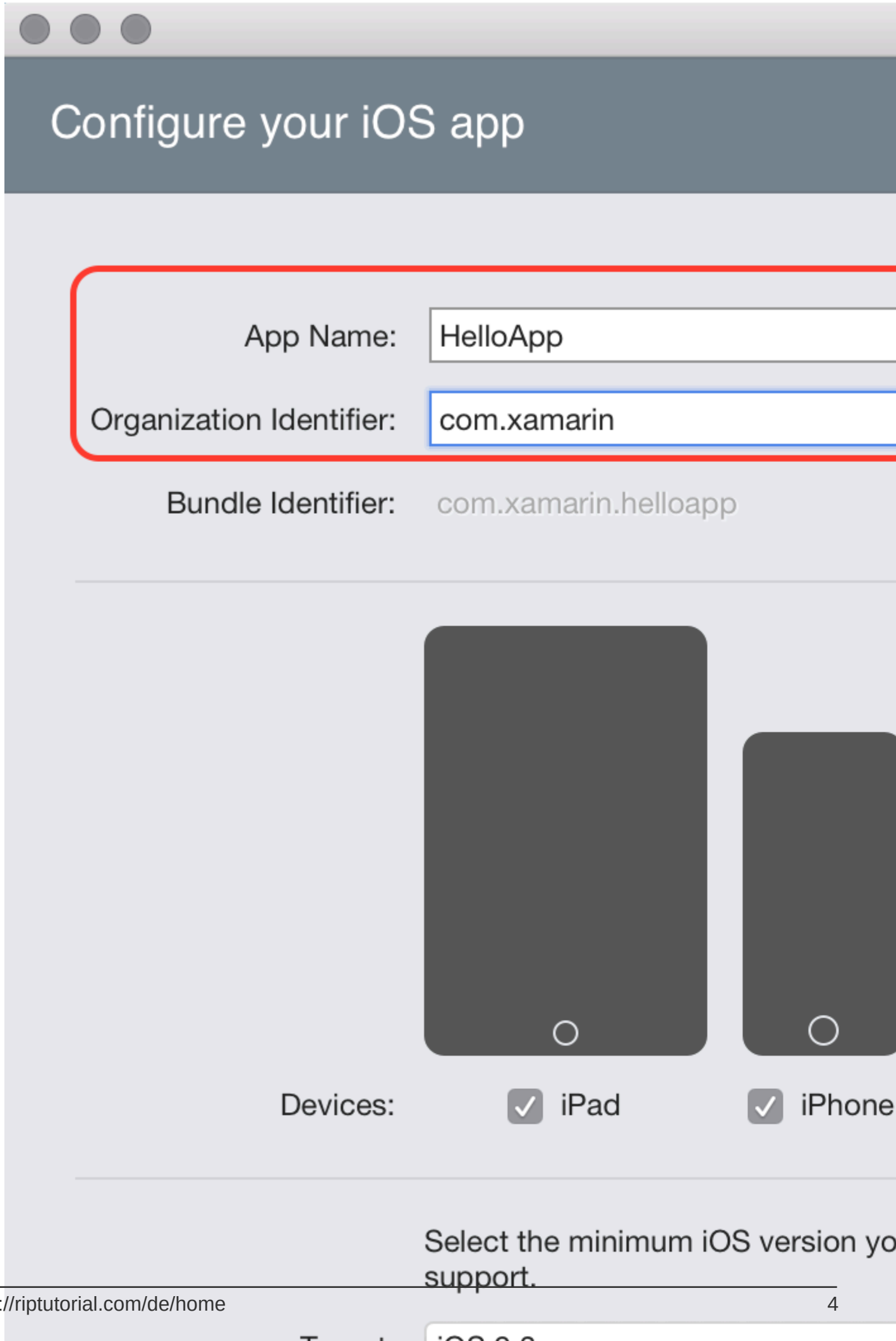
| Ausführung | Veröffentlichungsdatum |
|---|---|
| 1,0 | 2009-09-14 |
| 2,0 | 2010-04-05 |
| 3,0 | 2010-04-16 |
| 4,0 | 2011-04-06 |
| 5,0 | 2011-10-12 |
| 6,0 | 2012-09-19 |
| 7,0 | 2013-09-18 |
| 8,0 | 2014-09-10 |
| 9,0 | 2015-09-17 |
| 9.2 | 2015-11-17 |
| 9.4 | 2015-12-09 |
| 9.6 | 2016-03-22 |

Detaillierte Informationen zu jeder Version finden Sie hier:
https://developer.xamarin.com/releases/ios/

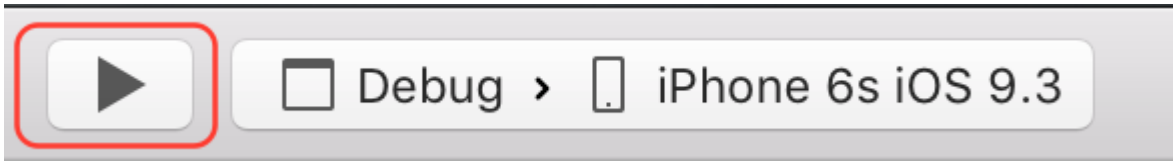
## Examples

### Starten Sie in Xamarin Studio

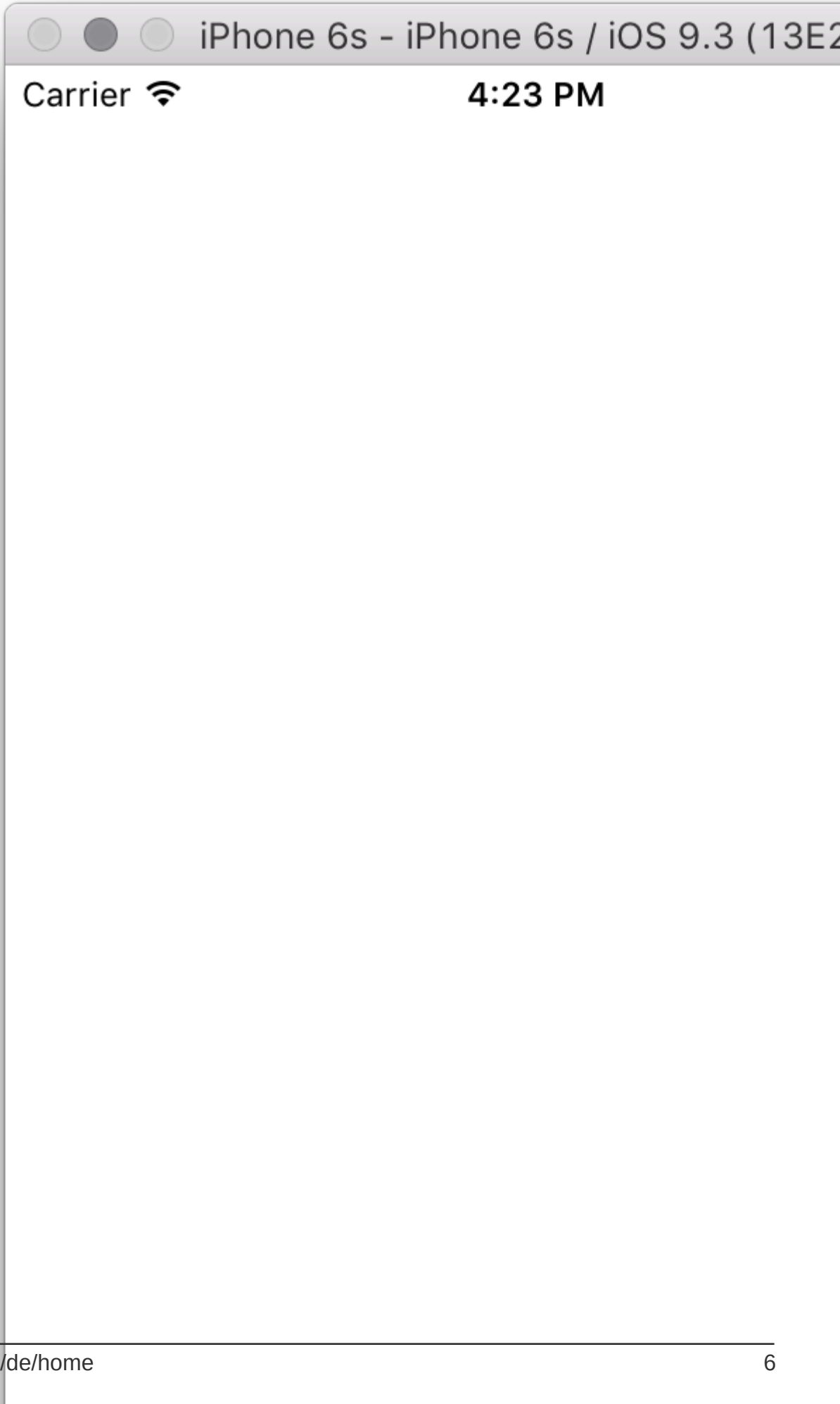
1. Navigieren Sie zu **Datei> Neu> Lösung** , um das Dialogfeld für das neue Projekt aufzurufen
2. Wählen Sie **Single View App** und drücken Sie **Next**
3. Konfigurieren Sie Ihre App, indem Sie Ihren App-Namen und Ihre Organisations-ID festlegen, und klicken Sie auf **Weiter** :


Configure your iOS app
App Name:
HelloApp
Organization Identifier:
com.xamarin
Bundle Identifier:
com.xamarin.helloapp
Devices:
iPad
iPhone
Select the minimum iOS version yo
support.

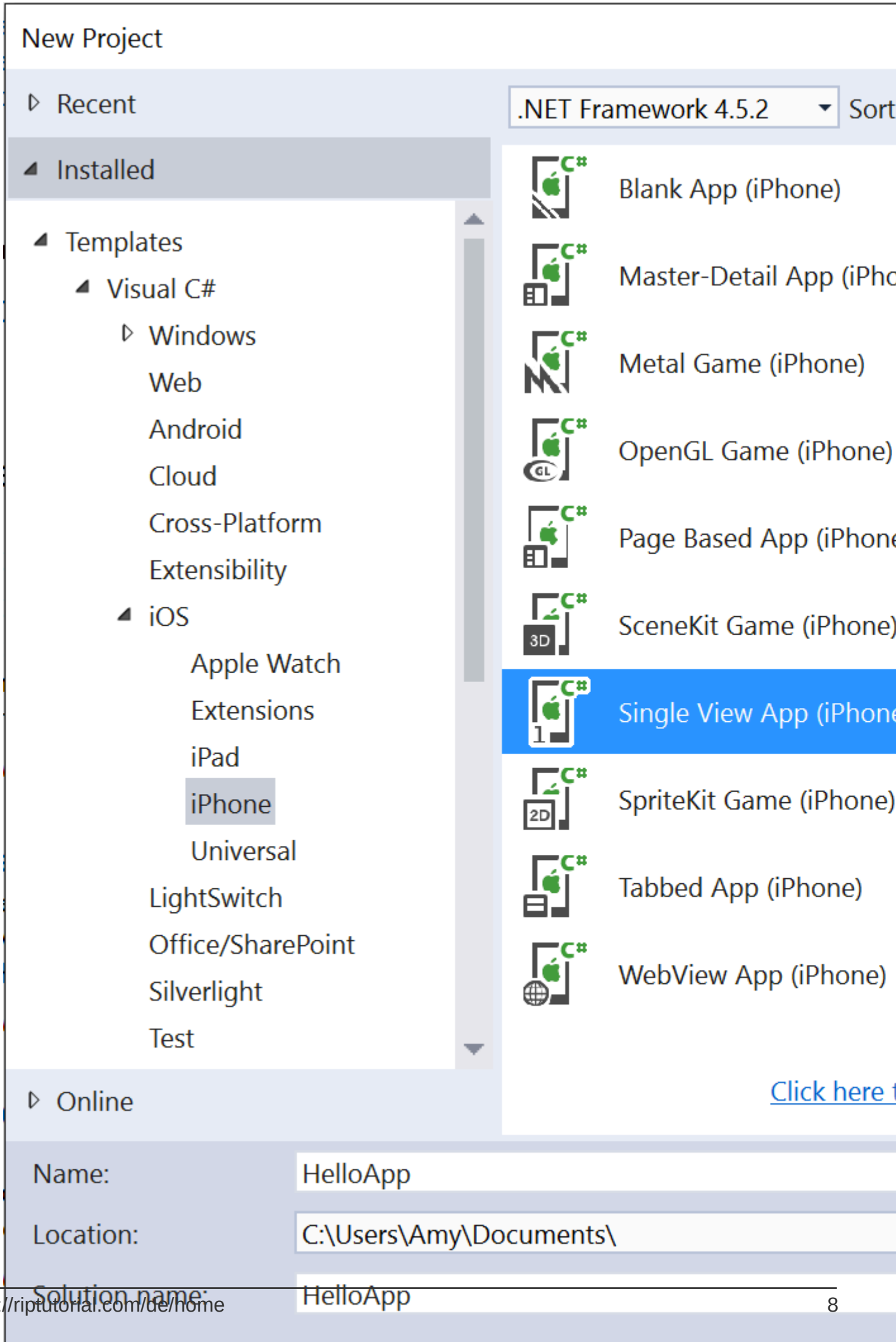
5. Um Ihre Anwendung auszuführen, wählen Sie Debug | Konfiguration von iPhone 6s iOS 9.x und drücken Sie die **Wiedergabetaste** :



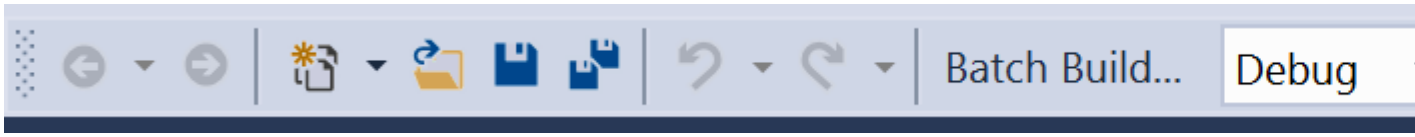

6. Dadurch wird der iOS-Simulator gestartet und die leere Anwendung angezeigt:


iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

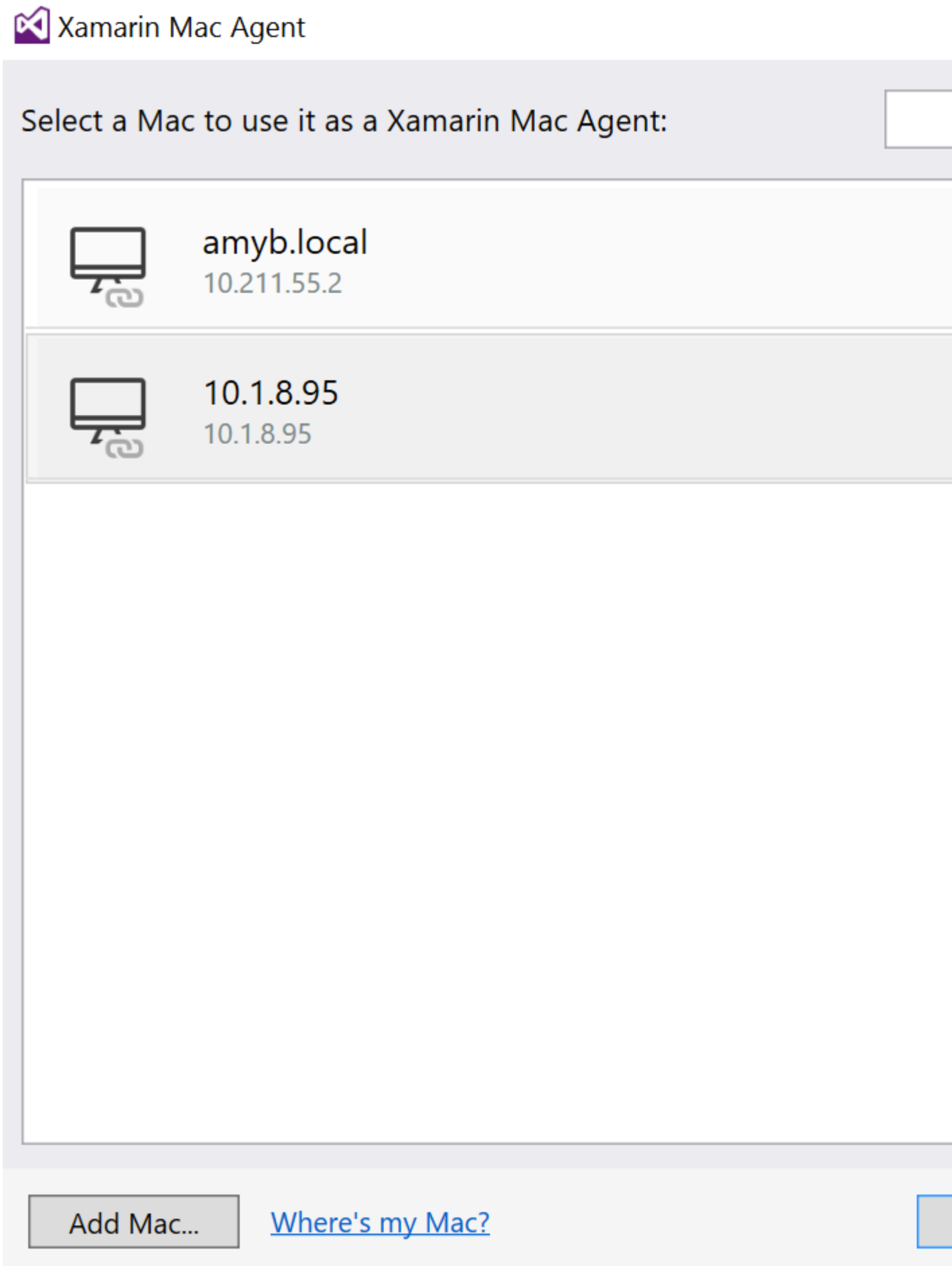
2. Navigieren Sie zu Visual C #> iOS> iPhone und wählen Sie Single View App:


New Project
Recent
.NET Framework 4.5.2
Sort
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
Apple Watch
Extensions
iPad
iPhone
Universal
LightSwitch
Office/SharePoint
Silverlight
Test
Online
Blank App (iPhone)
Master-Detail App (iPho
Metal Game (iPhone)
OpenGL Game (iPhone)
Page Based App (iPhone
SceneKit Game (iPhone)
Single View App (iPhone
SpriteKit Game (iPhone)
Tabbed App (iPhone)
WebView App (iPhone)
Click here t
Name:
HelloApp
Location:
C:\Users\Amy\Documents\
Solution name:
HelloApp

3. Geben Sie Ihrer App einen **Namen** und drücken Sie **OK** , um Ihr Projekt zu erstellen.
4. Wählen Sie das Mac Agent-Symbol in der Symbolleiste aus (siehe Abbildung unten):




5. Wählen Sie aus der Liste den Mac aus, der Ihre Anwendung erstellen soll (stellen Sie sicher, dass Ihr Mac für die Verbindung eingerichtet ist!), Und drücken Sie **Connect**

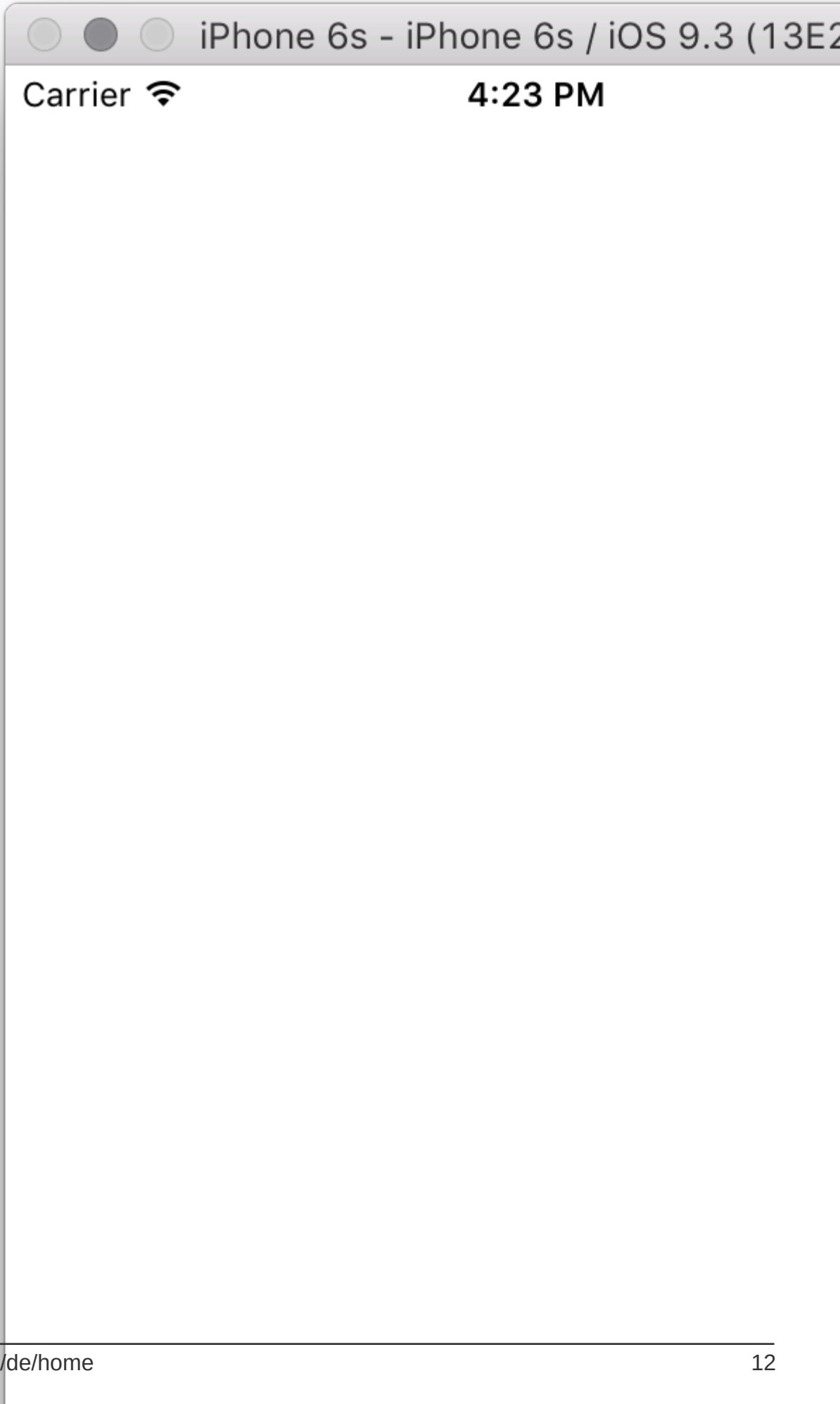



6. Um Ihre Anwendung auszuführen, wählen Sie **Debug | iPhoneSimulator-** Konfiguration und drücken Sie die Wiedergabetaste:

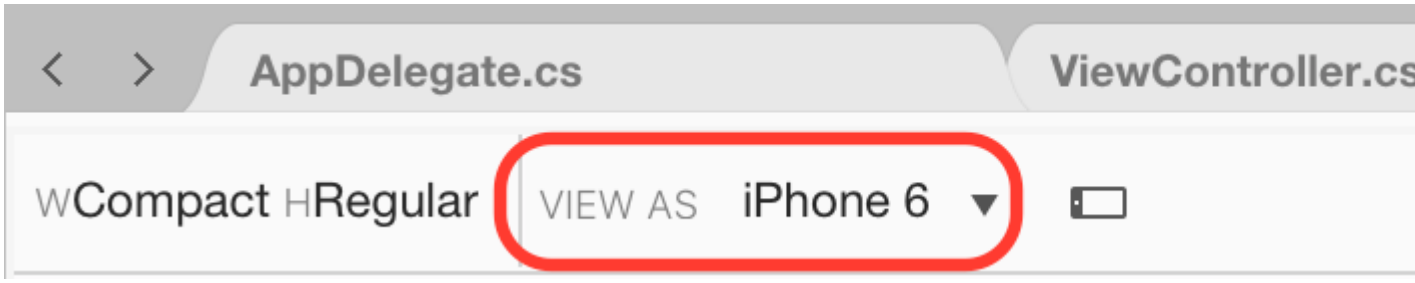
Konfiguration und drücken Sie die Wiedergabetaste:

Konfiguration und drücken Sie die Wiedergabetaste:

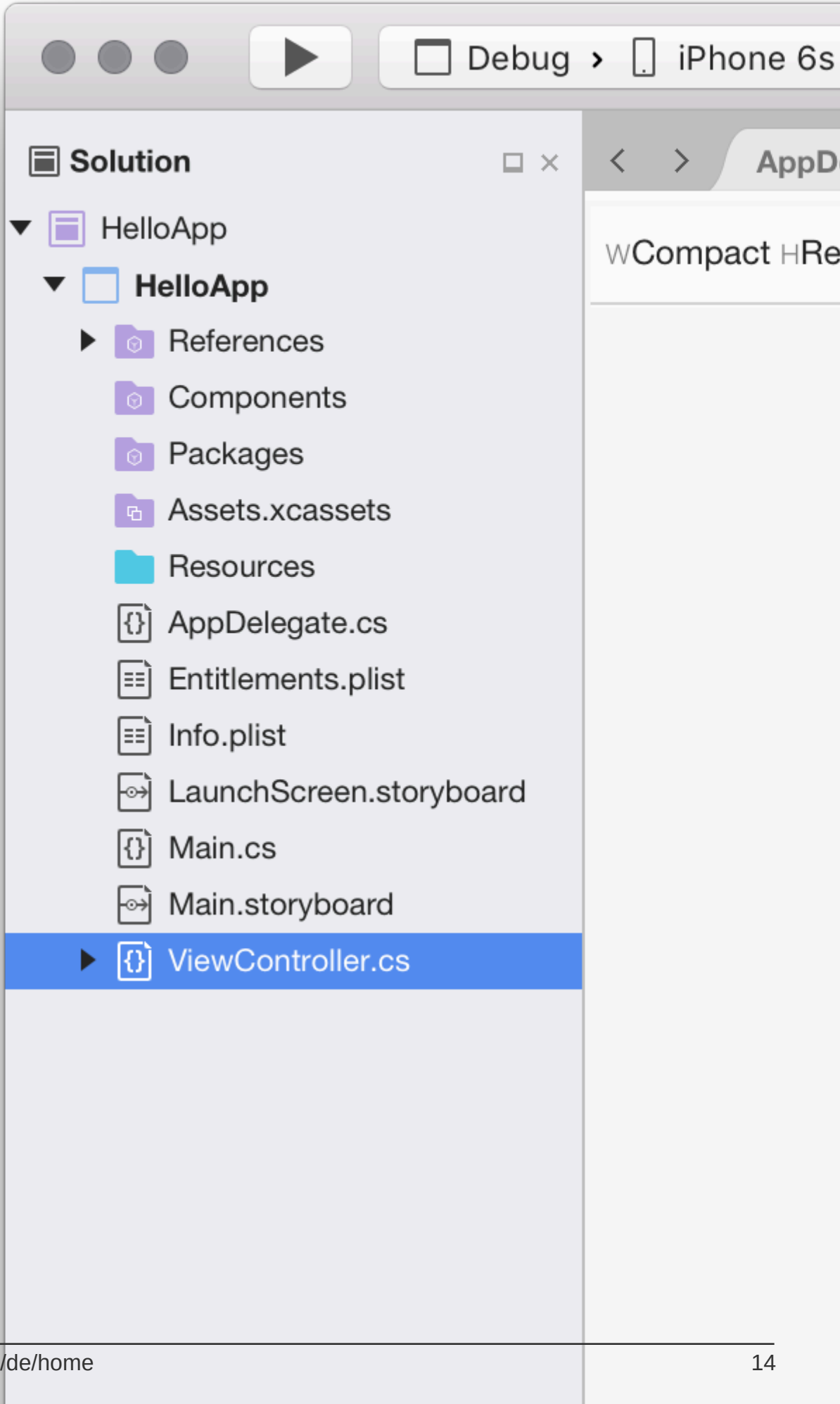

7. Dadurch wird der iOS-Simulator auf dem Mac gestartet und die leere Anwendung angezeigt:


iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

2. **Ansicht als** auf iPhone 6 einstellen:




3. Ziehen Sie eine Beschriftung und eine Schaltfläche aus der Toolbox auf die Entwurfsoberfläche, sodass sie wie das folgende Bild aussieht


Debug › iPhone 6s
Solution
HelloApp
HelloApp
References
Components
Packages
Assets.xcassets
Resources
AppDelegate.cs
Entitlements.plist
Info.plist
LaunchScreen.storyboard
Main.cs
Main.storyboard
ViewController.cs
AppD
wCompact HRe

4. Vergeben Sie im Eigenschaften-Pad dem Label und der Schaltfläche folgende Eigenschaften:

| nichts | Name | Titel |
|---|---|---|
| Etikette | lblClicks | [leer] |
| Taste | Klick mich | Klick mich! |

5. Fügen Sie der **ViewDidLoad-** Methode in der **ViewController-** Klasse den folgenden Code **hinzu** :

```
clickMe.TouchUpInside += (sender, e) =>
{
    totalClicks++;
    if (totalClicks == 1)
    {
        lblClicks.Text = totalClicks + " Click";
    }

   else {
       lblClicks.Text = totalClicks + " Clicks";
   }
};
```

6. Führen Sie die Anwendung aus

Erste Schritte mit Xamarin.iOS online lesen: https://riptutorial.com/de/xamarin-ios/topic/402/erste-schritte-mit-xamarin-ios

# Kapitel 2: Alarme

## Examples

### Alert anzeigen

Für Alarme seit iOS 8 würden Sie einen `UIAlertController` , für `UIAlertController` Versionen hätten Sie jedoch eine `UIAlertView` , die jetzt veraltet ist.

8,0

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.Alert);
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // otherTitle();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));
this.PresentViewController(alert, true, null);
```

8,0

```
var alert = new UIAlertView (title, message, null, cancelTitle, otherTitle);
alert.Clicked += (object sender, UIButtonEventArgs e) => {
    if(e.ButtonIndex == 1)
       // otherTitle();
};
alert.Show ();
```

### Zeigt eine Login-Warnung an

Der folgende Code gilt für iOS 8 und niedriger zum Erstellen einer Login-Warnung.

```
// Create the UIAlertView
var loginAlertView = new UIAlertView(title, message, null, cancelTitle, okTitle);

// Setting the AlertViewStyle to UIAlertViewStyle.LoginAndPasswordInput
loginAlertView.AlertViewStyle = UIAlertViewStyle.LoginAndPasswordInput;

// Getting the fields Username and Password
var usernameTextField = loginAlertView.GetTextField(0);
var passwordTextField = loginAlertView.GetTextField(1);

// Setting a placeholder
usernameTextField.Placeholder = "user@stackoverflow.com";
passwordTextField.Placeholder = "Password";

// Adding the button click handler.
loginAlertView.Clicked += (alertViewSender, buttonArguments) =>
{

    // Check if cancel button is pressed
    if (buttonArguments.ButtonIndex == loginAlertView.CancelButtonIndex)
    {
        // code
```

```
    }

    // In our case loginAlertView.FirstOtherButtonIndex is equal to the OK button
    if (buttonArguments.ButtonIndex == loginAlertView.FirstOtherButtonIndex)
    {
        // code
    }
};

// Show the login alert dialog
loginAlertView.Show();
```

## Aktionsblatt anzeigen

Mit `UIAlertController` seit `UIAlertController` verfügbaren UIAlertController können Sie dasselbe Alert-Objekt entweder für Aktionsblätter oder für klassische Alarme verwenden. Der einzige Unterschied ist der `UIAlertControllerStyle` , der beim Erstellen als Parameter übergeben wird.

Diese Zeile ändert sich von einer AlertView zu einem ActionSheet im Vergleich zu einigen anderen hier verfügbaren Beispielen:

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.ActionSheet);
```

Die Art und Weise, wie Sie dem Controller Aktionen hinzufügen, ist immer noch dieselbe:

```
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // ExecuteSomeAction();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));

//Add additional actions if necessary
```

Wenn Sie über eine parameterlose Void-Methode verfügen, können Sie diese als letzten Parameter der `.AddAction()` .

`private void DoStuff(){...}` wir zum Beispiel an, ich möchte, dass der Code von `private void DoStuff(){...}` ausgeführt wird, wenn ich "OK" drücke:

```
UIAlertAction action = UIAlertAction.Create("OK", UIAlertActionStyle.Cancel, DoStuff);
alert.AddAction(action);
```

Beachten Sie, dass ich bei der Erstellung der Aktion nicht das () nach DoStuff verwende.

Die Art und Weise, wie Sie den Controller präsentieren, erfolgt auf dieselbe Weise wie jeder andere Controller:

```
this.PresentViewController(alert, true, null);
```

## Dialog "Modal Alert" anzeigen

Es war allgemein üblich, `NSRunLoop` zum `NSRunLoop` von modalem `UIAlertView` zum Blockieren der

Codeausführung zu verwenden, bis Benutzereingaben in iOS verarbeitet werden. bis Apple das iOS7 veröffentlichte, brach es wenige vorhandene Apps. Glücklicherweise gibt es eine bessere Möglichkeit, es mit C # async / await zu implementieren.

Hier ist der neue Code, der das async / await-Muster nutzt, um modale UIAlertView zu zeigen:

```
Task ShowModalAletViewAsync (string title, string message, params string[] buttons)
{
    var alertView = new UIAlertView (title, message, null, null, buttons);
    alertView.Show ();
    var tsc = new TaskCompletionSource ();

    alertView.Clicked += (sender, buttonArgs) => {
        Console.WriteLine ("User clicked on {0}", buttonArgs.ButtonIndex);
        tsc.TrySetResult(buttonArgs.ButtonIndex);
    };
    return tsc.Task;
}

//Usage
async Task PromptUser() {
    var result = await ShowModalAletViewAsync
                ("Alert", "Do you want to continue?", "Yes", "No"); //process the result
}
```

Alarme online lesen: https://riptutorial.com/de/xamarin-ios/topic/433/alarme

# Kapitel 3: Arbeiten mit Xib und Storyboards in Xamarin.iOS

## Examples

### Öffnen Sie Xib / Sotryboard stattdessen im Xcode Interface Builder

Xamarin sturio öffnet Xib-Dateien und Storyboards standardmäßig im Xamarin-Designer. Benutzer kann mit der rechten Maustaste auf die Datei klicken und `Open With` -> `Xcode Interface Builder '

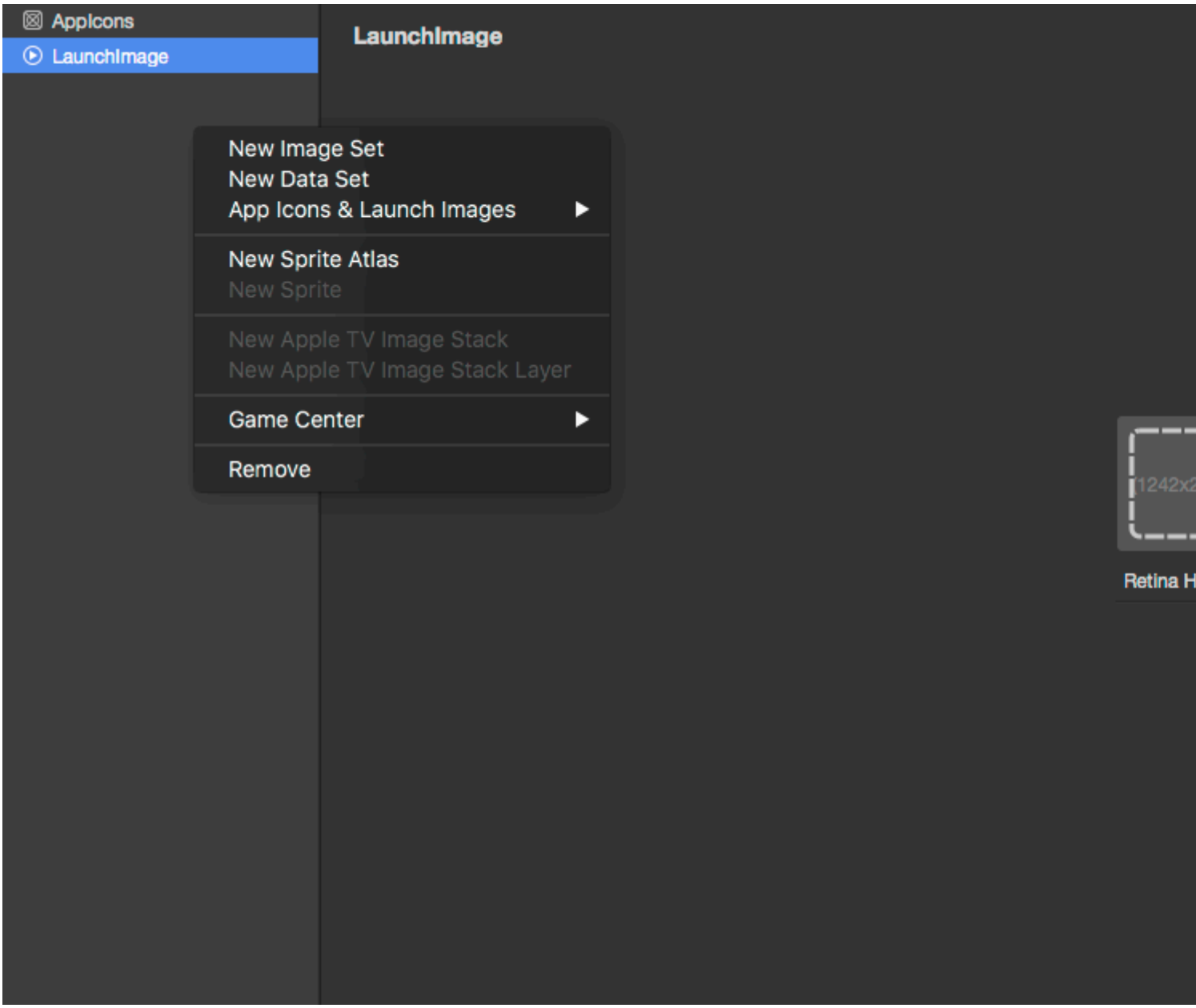
Arbeiten mit Xib und Storyboards in Xamarin.iOS online lesen: https://riptutorial.com/de/xamarin-ios/topic/6182/arbeiten-mit-xib-und-storyboards-in-xamarin-ios

# Kapitel 4: Asset-Kataloge verwenden

## Examples

### Hinzufügen von Image-Assets zum Asset-Katalog

So sieht der Asset-Katalog in Xamarin Studio aus:




Wie in der Abbildung oben gezeigt, gibt es fünf Arten von Assets, die Sie im Katalog erstellen können.

Ich werde nur das Bildset behandeln, weil es das einfachste ist.

Wenn Sie einen neuen Image-Satz erstellen. Sie werden Optionen wie diese erhalten


Um dem Katalog Bilder hinzuzufügen, klicken Sie einfach auf die gestrichelten Quadrate und

wählen Sie das Bild aus, das Sie für eine bestimmte Option festlegen möchten.

In XCode haben Sie die Optionen 1x, 2x und 3x, um die neuesten Bildschirmgrößen für iOS-Geräte abzudecken. Xamarin hat jedoch eine zusätzliche Option Vector, mit der Sie ein PDF-formatiertes Vektorbild hochladen können, das je nach Gerät, auf dem Ihre Anwendung ausgeführt wird, automatisch skaliert wird.

Für iPhone-Bilder behält Xamarin die iOS7-Spezialbildgröße R4 bei, die für iPhone mit 4-Zoll-Bildschirm (5, 5S und SE) verwendet wird.

Weitere Informationen finden Sie in der Xamarin-Dokumentation zum Hinzufügen von Bildern zur iOS-Anwendung .

Asset-Kataloge verwenden online lesen: https://riptutorial.com/de/xamarin-ios/topic/6630/asset-kataloge-verwenden

# Kapitel 5: Automatisches Layout in Xamarin.iOS

## Examples

### Hinzufügen von Einschränkungen mit iOS 9+ Layout-Ankern

9,0

```
// Since the anchor system simply returns constraints, you still need to add them somewhere.
View.AddConstraints(
    new[] {
        someLabel.TopAnchor.ConstraintEqualTo(TopLayoutGuide.GetBottomAnchor()),
        anotherLabel.TopAnchor.ConstraintEqualTo(someLabel.BottomAnchor, 6),
        oneMoreLabel.TopAnchor.ConstraintEqualTo(anotherLabel.BottomAnchor, 6),

oneMoreLabel.BottomAnchor.ConstraintGreaterThanOrEqualTo(BottomLayoutGuide.GetTopAnchor(), -
10),
    }
);
```

### Hinzufügen von Einschränkungen mit der Visual Format Language (VFL)

```
// Using Visual Format Language requires a special look-up dictionary of names<->views.
var views = new NSDictionary(
    nameof(someLabel), someLabel,
    nameof(anotherLabel), anotherLabel,
    nameof(oneMoreLabel), oneMoreLabel
);
// It can also take a look-up dictionary for metrics (such as size values).
// Since we are hard-coding those values in this example, we can give it a `null` or empty
dictionary.
var metrics = (NSDictionary)null;

// Add the vertical constraints to stack everything together.
// `V:` = vertical
// `|…|` = constrain to super view (`View` for this example)
// `-10-` = connection with a gap of 10 pixels (could also be a named parameter from the
metrics dictionary)
// `-[viewName]-` = connection with a control by name looked up in views dictionary (using C#
6 `nameof` for refactoring support)
var verticalConstraints = NSLayoutConstraint.FromVisualFormat(
    $"V:|-20-[{nameof(someLabel)}]-6-[{nameof(anotherLabel)}]-6-[{nameof(oneMoreLabel)}]->=10-
|",
    NSLayoutFormatOptions.AlignAllCenterX,
    metrics,
    views
);
View.AddConstraints(verticalConstraints);
```

Möglicherweise finden Sie einige Einschränkungstypen, z. B. Seitenverhältnisse , nicht in der VFL-Syntax (Visual Format Language) und müssen die entsprechenden Methoden direkt aufrufen.

## Verwendung von Cirrious.FluentLayout

NuGet verwenden

```
Install-Package Cirrious.FluentLayout
```

Ein erweitertes Beispiel basierend auf dem Starter-Beispiel auf der GitHub- Seite, ein einfacher Vorname, Nachname-Beschriftungen und Felder, die alle aufeinander gestapelt sind:

```
public override void ViewDidLoad()
{
    //create our labels and fields
    var firstNameLabel = new UILabel();
    var lastNameLabel = new UILabel();
    var firstNameField = new UITextField();
    var lastNameField = new UITextField();

    //add them to the View
    View.AddSubviews(firstNameLabel, lastNameLabel, firstNameField, lastNameField);

    //create constants that we can tweak if we do not like the final layout
    const int vSmallMargin = 5;
    const int vMargin = 20;
    const int hMargin = 10;

    //add our constraints
    View.SubviewsDoNotTranslateAutoresizingMaskIntoConstraints();
    View.AddConstraints(
        firstNameLabel.WithSameTop(View).Plus(vMargin),
        firstNameLabel.AtLeftOf(View).Plus(hMargin),
        firstNameLabel.WithSameWidthOf(View),

        firstNameField.WithSameWidth(firstNameLabel),
        firstNameField.WithSameLeft(firstNameLabel),
        firstNameField.Below(firstNameLabel).Plus(vSmallMargin),

        lastNameLabel.Below(firstNameField).Plus(vMargin),
        lastNameLabel.WithSameLeft(firstNameField),
        lastNameLabel.WithSameWidth(firstNameField),

        lastNameField.Below(lastNameLabel).Plus(vSmallMargin),
        lastNameField.WithSameWidth(lastNameLabel),
        lastNameField.WithSameLeft(lastNameLabel));
}
```

## Hinzufügen von Einschränkungen mit Mauerwerk

Masonry ist eine Bibliothek für Objective-c, aber Xamarin hat dafür eine Bindung erstellt und als Nuget-Paket https://www.nuget.org/packages/Masonry/ erstellt .

Nuget installieren

```
Install-Package Masonry
```

Dies zentriert eine Schaltfläche 100 Punkte unter dem Mittelpunkt der enthaltenden Ansicht und

legt eine Breite zwischen 200 und 400 Punkten fest

```
this.loginBtn.MakeConstraints(make =>
{
    make.Width.GreaterThanOrEqualTo(new NSNumber(200));
    make.Width.LessThanOrEqualTo(new NSNumber(400));
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, 100));
});
```

Dadurch wird ein skaliertes Bild um 100 Punkte oberhalb des Mittelpunkts der enthaltenden Ansicht festgelegt. Dann wird die Breite mit einem Multiplikator von 0,5 auf die Breite der enthaltenden Ansicht festgelegt, dh 50% der Breite. Dann wird die Höhe auf die Breite multipliziert mit dem Seitenverhältnis gesetzt, wodurch das Bild skaliert wird, das korrekte Seitenverhältnis jedoch beibehalten wird

```
this.logo.MakeConstraints(make =>
{
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, -100));
    make.Width.EqualTo(this.View).MultipliedBy(0.5f);
    make.Height.EqualTo(this.logo.Width()).MultipliedBy(0.71f);
});
```

Automatisches Layout in Xamarin.iOS online lesen: https://riptutorial.com/de/xamarin-ios/topic/1317/automatisches-layout-in-xamarin-ios

# Kapitel 6: Berechnung der variablen Zeilenhöhe in GetHeightForRow

## Bemerkungen

Das Berechnen von Zeilenhöhen ist möglicherweise teuer und die Bildlaufleistung kann beeinträchtigt werden, wenn Sie über größere Datenmengen verfügen. Überschreiben `UITableViewSource.EstimatedHeight(UITableView, NSIndexPath)` in diesem Szenario `UITableViewSource.EstimatedHeight(UITableView, NSIndexPath)` um schnell eine für das schnelle Scrollen ausreichende Zahl bereitzustellen, z. B .:

```
public override nfloat EstimatedHeight(UITableView tableView, NSIndexPath indexPath)
{
    return 44.0f;
}
```

## Examples

### GetHeightForRow verwenden

Überschreiben Sie `UITableViewSource.GetHeightForRow(UITableView,NSIndexPath)` um eine benutzerdefinierte `UITableViewSource.GetHeightForRow(UITableView,NSIndexPath)`

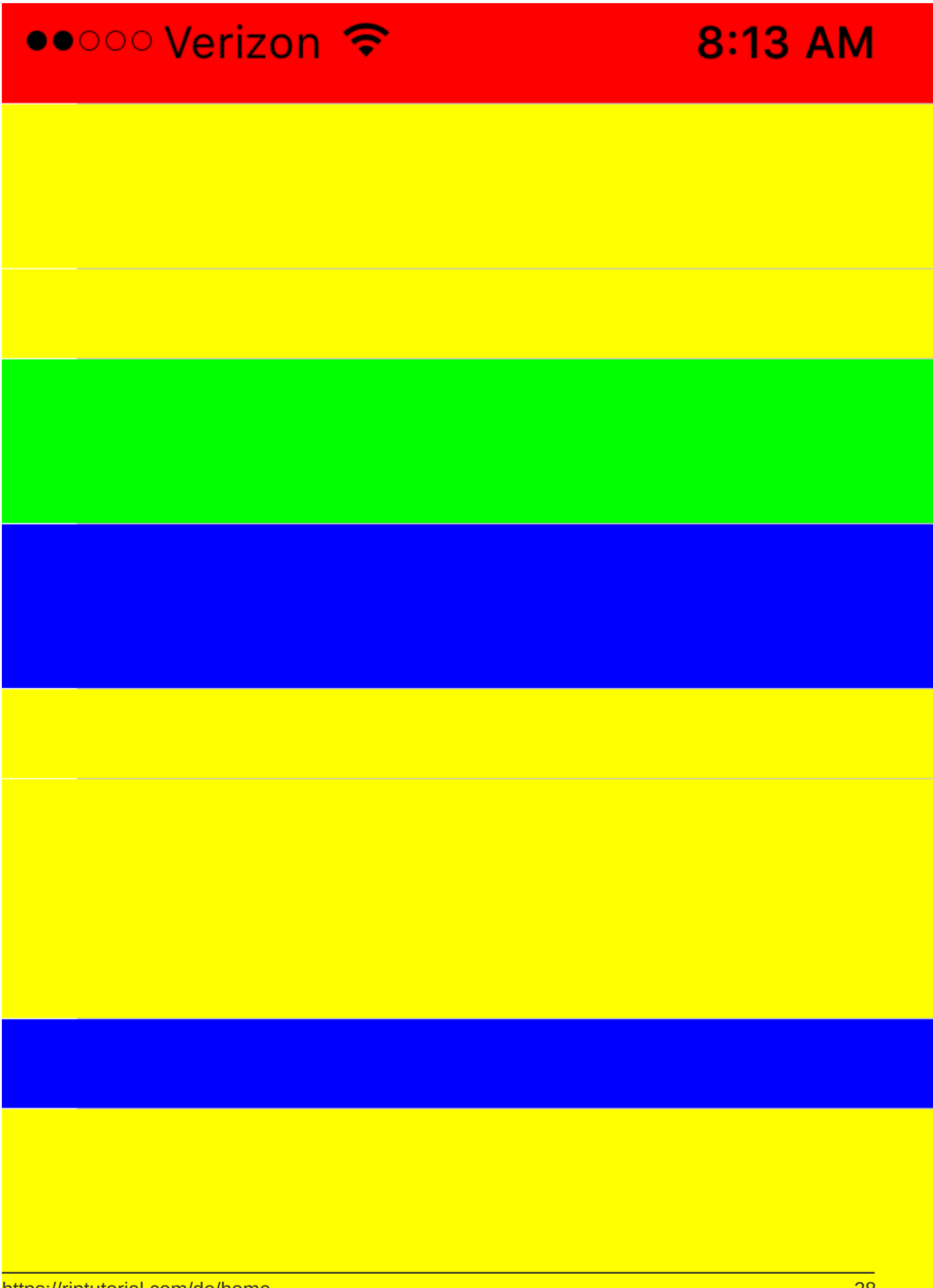
```
public class ColorTableDataSource : UITableViewSource
{
    List<DomainClass> Model { get; set; }

    public override nfloat GetHeightForRow(UITableView tableView, NSIndexPath indexPath)
    {
        var height = Model[indexPath.Row % Model.Count].Height;
        return height;
    }

    //...etc ...
}
```

Die Domänenklasse für die Tabelle (in diesem Fall hat sie 1 von 3 zufälligen Farben und 1 von 3 zufälligen Höhen):

```
public class DomainClass
{
    static Random rand = new Random(0);
    public UIColor Color { get; protected set; }
    public float Height { get; protected set; }

    static UIColor[] Colors = new UIColor[]
    {
        UIColor.Red,
```

```
        UIColor.Green,
        UIColor.Blue,
        UIColor.Yellow
    };

    public DomainClass()
    {
        Color = Colors[rand.Next(Colors.Length)];
        switch (rand.Next(3))
        {
            case 0:
                Height = 24.0f;
                break;
            case 1:
                Height = 44.0f;
                break;
            case 2:
                Height = 64.0f;
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }
    }

    public override string ToString()
    {
        return string.Format("[DomainClass: Color={0}, Height={1}]", Color, Height);
    }
}
```

Was aussieht wie:


Verizon
8:13 AM

https://riptutorial.com/de/xamarin-ios/topic/6515/berechnung-der-variablen-zeilenhohe-in-getheightforrow

# Kapitel 7: Berührungsidentifikation

## Parameter

| Säule | Säule |
|---|---|
| Zelle | Zelle |

## Bemerkungen

Stellen Sie zunächst fest, ob das Gerät die Touch ID-Eingabe akzeptieren kann.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Wenn dies der Fall ist, können Sie die Touch-ID-Benutzeroberfläche anzeigen, indem Sie Folgendes verwenden:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Es gibt drei Informationen, die wir an `EvaluatePolicy` - die Richtlinie selbst, eine Zeichenfolge, aus der hervorgeht, warum eine Authentifizierung erforderlich ist, und einen Antworthandler. Der Antworthandler teilt der Anwendung mit, was sie im Falle einer erfolgreichen oder erfolglosen Authentifizierung tun soll.

Eine der Einschränkungen der lokalen Authentifizierung besteht darin, dass sie im Vordergrund ausgeführt werden muss. `InvokeOnMainThread` daher sicher, dass Sie `InvokeOnMainThread` für den `InvokeOnMainThread` :

```
var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
```

Um festzustellen, ob die Datenbank autorisierter Fingerabdrücke geändert wurde, können Sie die

von `context.EvaluatedPolicyDomainState` undurchsichtige Struktur (NSData) `context.EvaluatedPolicyDomainState` . Ihre App muss den Richtlinienstatus speichern und vergleichen, um Änderungen zu erkennen. Eine Sache, die Apple feststellt:

> Die Art der Änderung kann jedoch nicht aus diesen Daten bestimmt werden.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

# Examples

## Fügen Sie Ihrer App eine Touch-ID hinzu

Stellen Sie zunächst fest, ob das Gerät die Touch ID-Eingabe akzeptieren kann.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Wenn dies der Fall ist, können Sie die Touch-ID-Benutzeroberfläche anzeigen, indem Sie Folgendes verwenden:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Es gibt drei Informationen, die wir an `EvaluatePolicy` - die Richtlinie selbst, eine Zeichenfolge, aus der hervorgeht, warum eine Authentifizierung erforderlich ist, und einen Antworthandler. Der Antworthandler teilt der Anwendung mit, was sie im Falle einer erfolgreichen oder erfolglosen Authentifizierung tun soll.

Eine der Einschränkungen der lokalen Authentifizierung besteht darin, dass sie im Vordergrund

ausgeführt werden muss. `InvokeOnMainThread` daher sicher, dass Sie `InvokeOnMainThread` für den `InvokeOnMainThread` :

```
var replyHandler = new LAContextReplyHandler((success, error) =>
{
    this.InvokeOnMainThread(() =>
    {
        if (success)
        {
            Console.WriteLine("You logged in!");
            PerformSegue("AuthenticationSegue", this);
        }
        else {
            //Show fallback mechanism here
        }
    });
});
```

Um festzustellen, ob die Datenbank autorisierter Fingerabdrücke geändert wurde, können Sie die von `context.EvaluatedPolicyDomainState` undurchsichtige Struktur (NSData) `context.EvaluatedPolicyDomainState` . Ihre App muss den Richtlinienstatus speichern und vergleichen, um Änderungen zu erkennen. Eine Sache, die Apple feststellt:

> Die Art der Änderung kann jedoch nicht aus diesen Daten bestimmt werden.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

### Schaltflächenbeispiel

```
partial void AuthenticateMe(UIButton sender)
{
    var context = new LAContext();
    //Describes an authentication context
```

```
    //that allows apps to request user authentication using Touch ID.
    NSError AuthError;
    //create the reference for error should it occur during the authentication.
    var myReason = new NSString("To add a new chore");
    //this is the string displayed at the window for touch id

    if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
    // check if the device have touchId capabilities.
    {
        var replyHandler = new LAContextReplyHandler((success, error) =>
        {

            this.InvokeOnMainThread(() =>
            {
                if (success)
                {
                    Console.WriteLine("You logged in!");
                    PerformSegue("AuthenticationSegue", this);
                }
                else {
                    //Show fallback mechanism here
                }
            });

        });
        context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);//send touch id request
    };
}
```

## Schlüsselbund verwenden

Arbeitsquelle - https://github.com/benhysell/V.TouchIdExample

Langformbeschreibung - http://benjaminhysell.com/archive/2014/11/authentication-in-xamarin-ios-with-touch-id-or-passcode/

```
//Simple View with a switch to enable / disable Touch ID and
//a button to invoke authentication

/// <summary>
/// Enable/Disable Touch ID
/// </summary>
/// <param name="sender">Sender.</param>
partial void TouchIdEnableDisable(UISwitch sender)
{
    if (sender.On)
    {
        //enable Touch ID
        //set our record
        //note what you fill in here doesn't matter, just needs to be
        //consistent across all uses of the record
        var secRecord = new SecRecord(SecKind.GenericPassword)
        {
            Label = "Keychain Item",
            Description = "fake item for keychain access",
            Account = "Account",
```

```
            Service = "com.yourcompany.touchIdExample",
            Comment = "Your comment here",
            ValueData = NSData.FromString("my-secret-password"),
            Generic = NSData.FromString("foo")
        };

        secRecord.AccessControl = new
SecAccessControl(SecAccessible.WhenPasscodeSetThisDeviceOnly,
SecAccessControlCreateFlags.UserPresence);
        SecKeyChain.Add(secRecord);

        authenticateButton.Enabled = true;
    }
    else
    {
        //disable Touch ID
        var record = new SecRecord(SecKind.GenericPassword)
        {
            Service = "com.yourcompany.touchIdExample",
            UseOperationPrompt = "Authenticate to Remove Touch ID / Passcode from Test App"
        };

        SecStatusCode result;

        //query one last time to ensure they can remove it
        SecKeyChain.QueryAsRecord(record, out result);
        if (SecStatusCode.Success == result || SecStatusCode.ItemNotFound == result)
        {
            //remove the record
            SecKeyChain.Remove(record);
            authenticateButton.Enabled = false;
        }
        else
        {
            //could not authenticate, leave switch on
            sender.On = true;
         }
    }
}

/// <summary>
/// Show Touch ID to user and evaluate authentication
/// </summary>
/// <param name="sender">Sender.</param>
partial void AuthenticateUser(UIButton sender)
{
    var rec = new SecRecord(SecKind.GenericPassword)
    {
        Service = "com.yourcompany.touchIdExample",
        UseOperationPrompt = "Authenticate to access Test App"
    };
    SecStatusCode res;
    SecKeyChain.QueryAsRecord(rec, out res);
    if (SecStatusCode.Success == res || SecStatusCode.ItemNotFound == res)
    {
        //Success!!
        //add your code here to continue into your application
        AuthenticatedLabel.Hidden = false;
    }
    else
    {
```

```
        //Failure
        AuthenticatedLabel.Hidden = true;
    }
}
```

Berührungsidentifikation online lesen: https://riptutorial.com/de/xamarin-ios/topic/577/beruhrungsidentifikation

# Kapitel 8: Best Practices für die Migration von UILocalNotification zu User Notifications Framework

## Examples

### Benutzerbenachrichtigungen

1. Sie müssen UserNotifications importieren

```
@import UserNotifications;
```

2. Erlaubnis für localNotification anfordern

```
let center = UNUserNotificationCenter.current()
center.requestAuthorization([.alert, .sound]) { (granted, error) in
    // Enable or disable features based on authorization.
}
```

3. Jetzt aktualisieren wir die Anwendungssymbol-Ausweisnummer

```
 @IBAction  func triggerNotification(){
 let content = UNMutableNotificationContent()
 content.title = NSString.localizedUserNotificationString(forKey: "Tom said:", arguments:
nil)
 content.body = NSString.localizedUserNotificationString(forKey: "Hello Mike Let's go.",
arguments: nil)
 content.sound = UNNotificationSound.default()
 content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
 content.categoryIdentifier = "com.mike.localNotification"
 //Deliver the notification in two seconds.
 let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 1.0, repeats: true)
 let request = UNNotificationRequest.init(identifier: "TwoSecond", content: content,
trigger: trigger)

 //Schedule the notification.
 let center = UNUserNotificationCenter.current()
 center.add(request)
 }

 @IBAction func stopNotification(_ sender: AnyObject) {
 let center = UNUserNotificationCenter.current()
 center.removeAllPendingNotificationRequests()
 }
```

Best Practices für die Migration von UILocalNotification zu User Notifications Framework online lesen: https://riptutorial.com/de/xamarin-ios/topic/6382/best-practices-fur-die-migration-von-uilocalnotification-zu-user-notifications-framework

# Kapitel 9: Binden Sie schnelle Bibliotheken

## Einführung

Eine einfach zu befolgende Anleitung, die Sie durch den Prozess der Bindung von Swift .framework-Dateien für die Verwendung in einem Xamarin-Projekt führt.

## Bemerkungen

1. Wenn Sie eine Bibliothek in Xcode erstellen, haben Sie die Möglichkeit, die schnellen Bibliotheken einzubinden. Nicht! Sie werden in Ihre endgültige App als NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib aufgenommen, müssen jedoch als NAME.app/Frameworks/libswift*.dylib enthalten sein

2. Sie finden diese Informationen an anderer Stelle, aber es ist erwähnenswert: Fügen Sie Bitcode nicht in die Bibliothek ein. Im Moment enthält Xamarin Bitcode für iOS nicht und Apple erfordert, dass alle Bibliotheken die gleichen Architekturen unterstützen.

## Examples

### Binden einer schnellen Bibliothek in Xamarin.iOS

Das Binden einer schnellen Bibliothek in Xamarin.iOS erfolgt für Objective-C auf dieselbe Weise wie in https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/ beschrieben , jedoch mit einigen Vorbehalten.

1. Eine schnelle Klasse muss von NSObject erben, um gebunden zu werden.
2. Der Swift-Compiler übersetzt Klassen- und Protokollnamen in etwas anderes, sofern Sie nicht die @objc- Annotation (z. B. @objc (MyClass)) in Ihren Swift-Klassen verwenden, um den expliziten Ziel-c-Namen anzugeben.
3. Zur Laufzeit muss Ihre APP neben Ihrem gebundenen Framework einige schnelle Kernbibliotheken in einem Ordner namens Frameworks enthalten.
4. Wenn die App in AppStore gepusht wird, muss sie neben Ihrem Payload-Ordner einen SwiftSupport-Ordner enthalten. Diese befinden sich in der IPA-Datei.

Hier finden Sie eine einfache Musterbindung:
https://github.com/Flash3001/Xamarin.BindingSwiftLibrarySample

Und ein vollständiges Bindungsbeispiel: https://github.com/Flash3001/iOSCharts.Xamarin

Schritte finden Sie unten:

# 1.1 Bereiten Sie die Swift-Klassen vor, die Sie

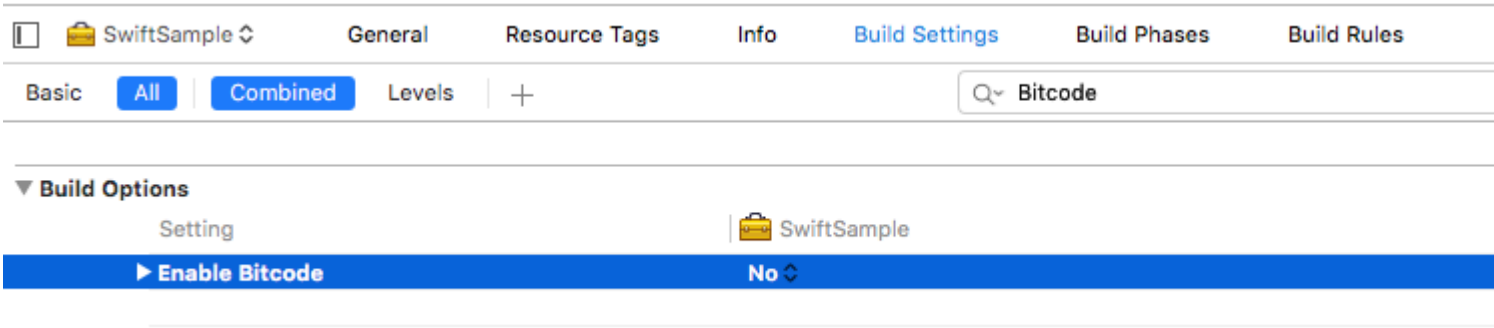
# exportieren möchten

Für jede Swift-Klasse, die Sie verwenden möchten, müssen Sie entweder von NSObject erben und den Objective-C-Namen mithilfe der objc-Annotation explizit machen. Andernfalls generiert der Swift-Compiler andere Namen. Unten ist ein Beispielcode, wie eine Swift-Klasse aussehen könnte. Beachten Sie, dass es egal ist, welche Klasse er übernimmt, solange die Stammklasse von NSObject erbt.

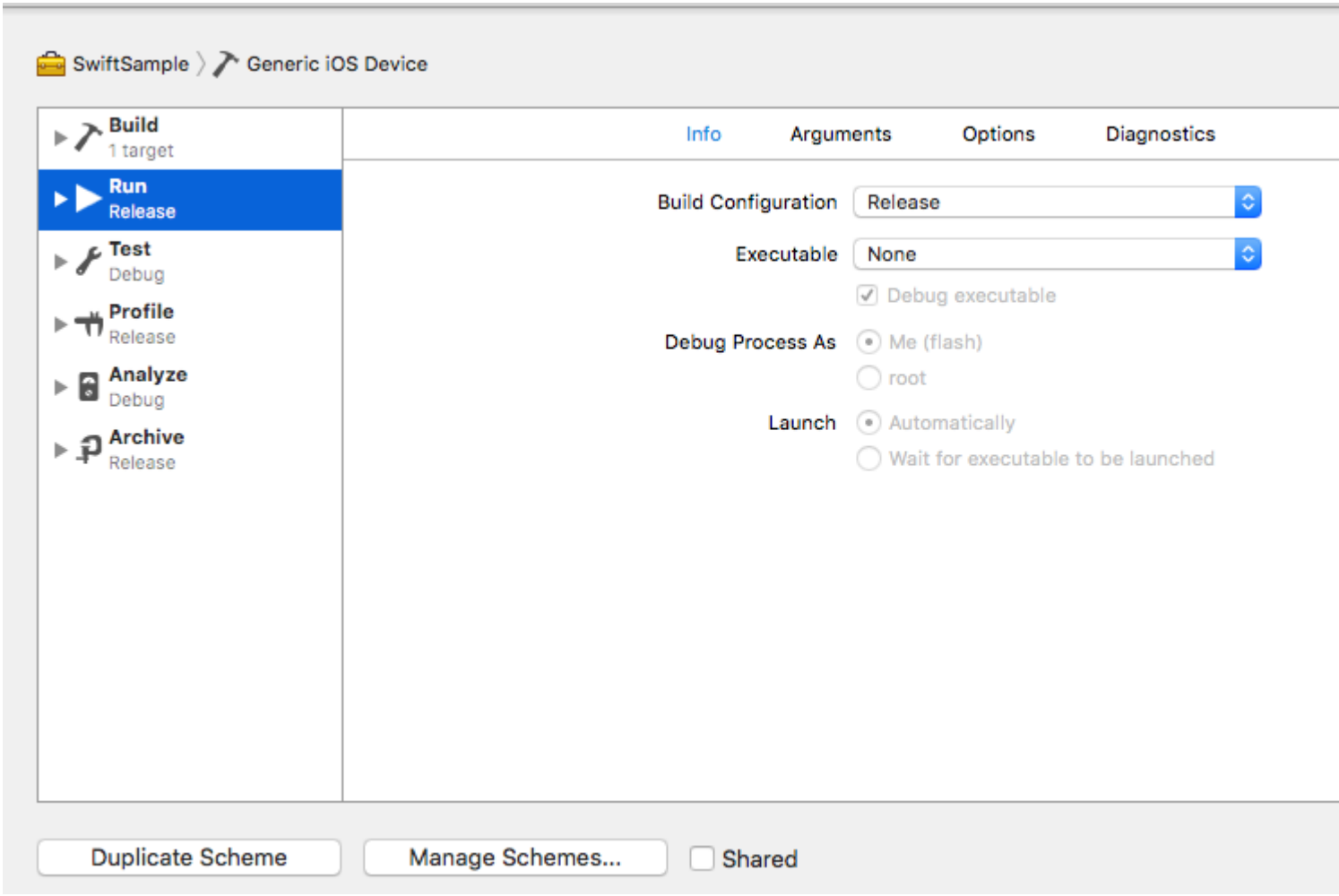
```
//Add this to specify explicit objective c name
@objc(MyClass)
open class MyClass: NSObject {
    open func getValue() -> String
    {
        return "Value came from MyClass.swift!";
    }
}
```

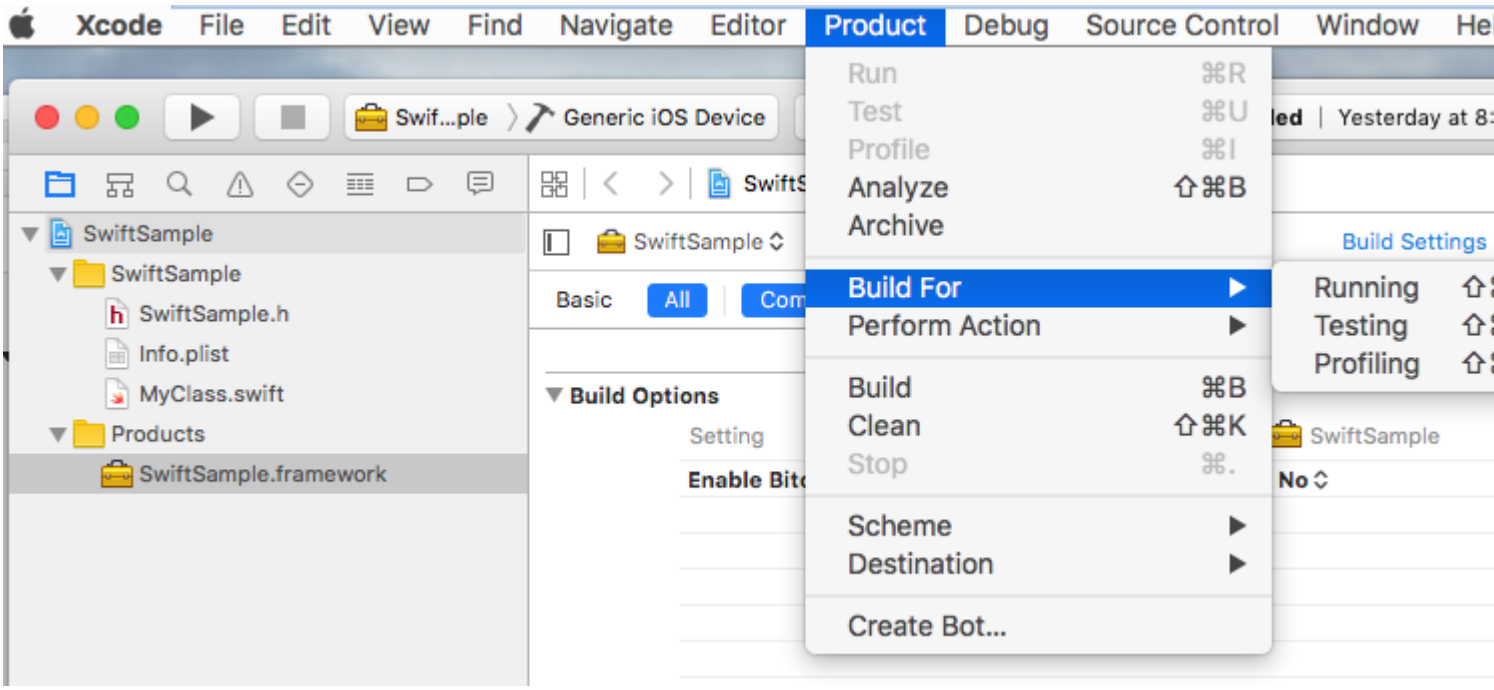
## 1.2 Bauen Sie den Rahmen auf

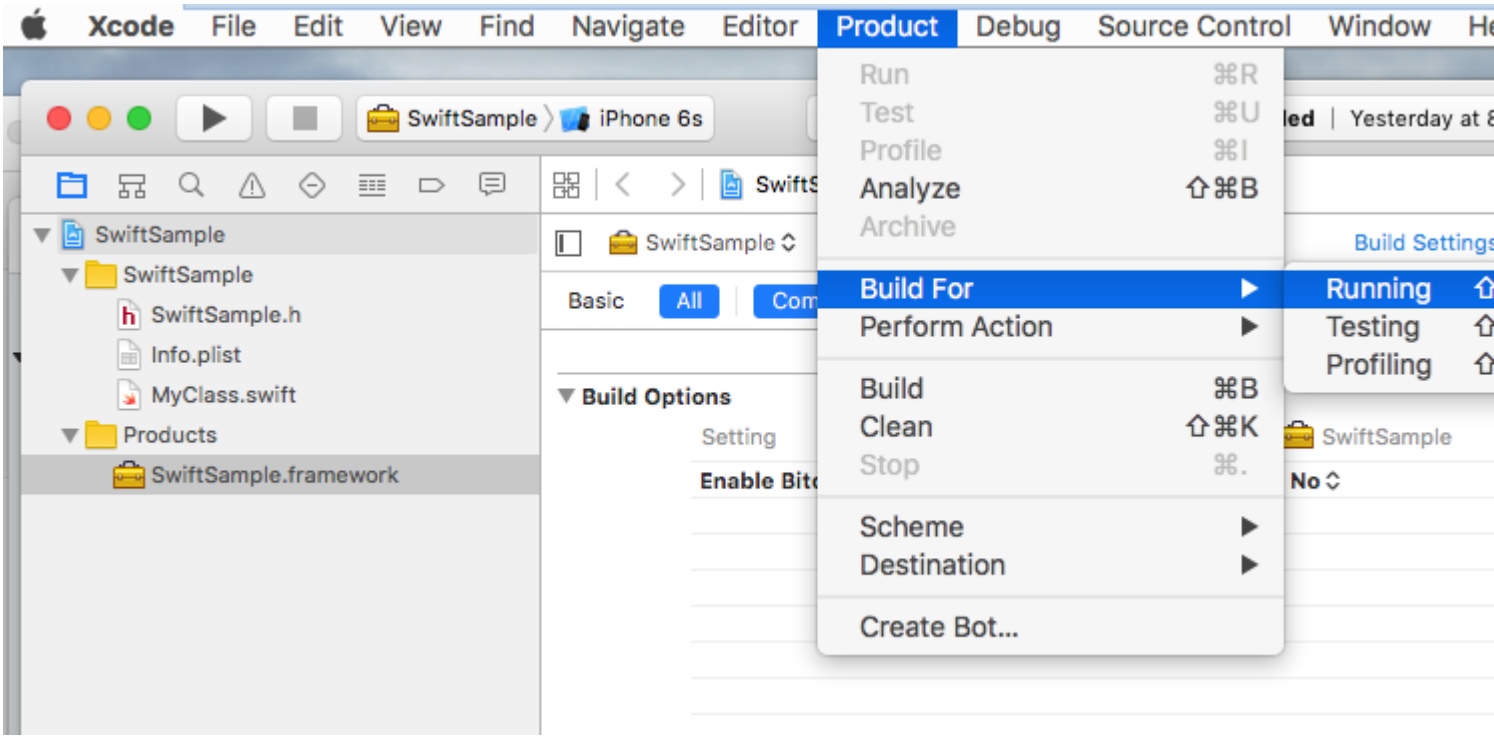
Bitcode deaktivieren *

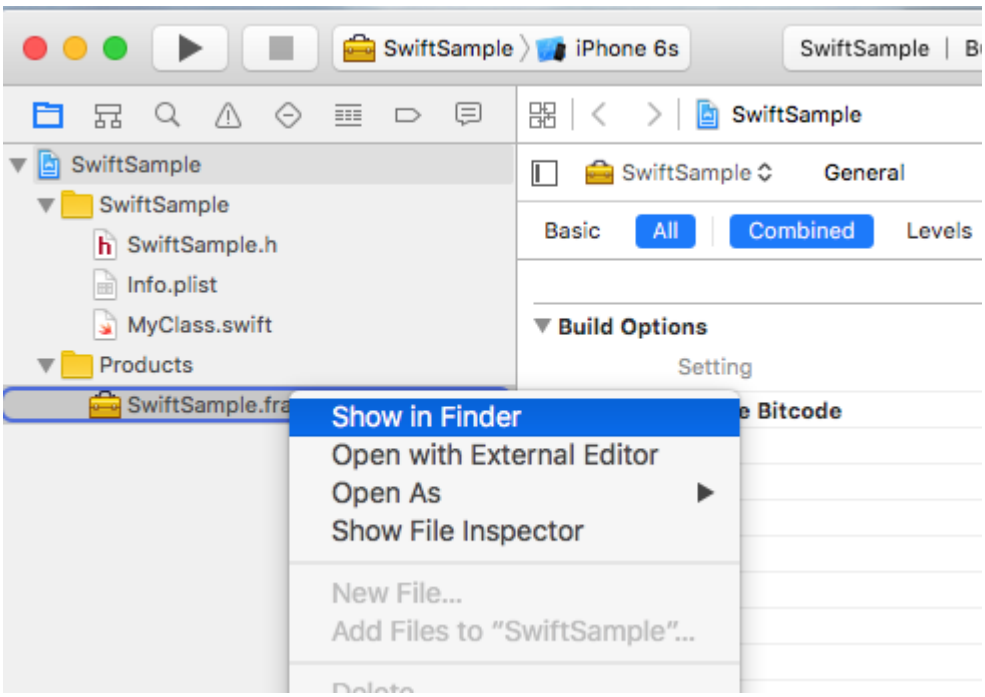


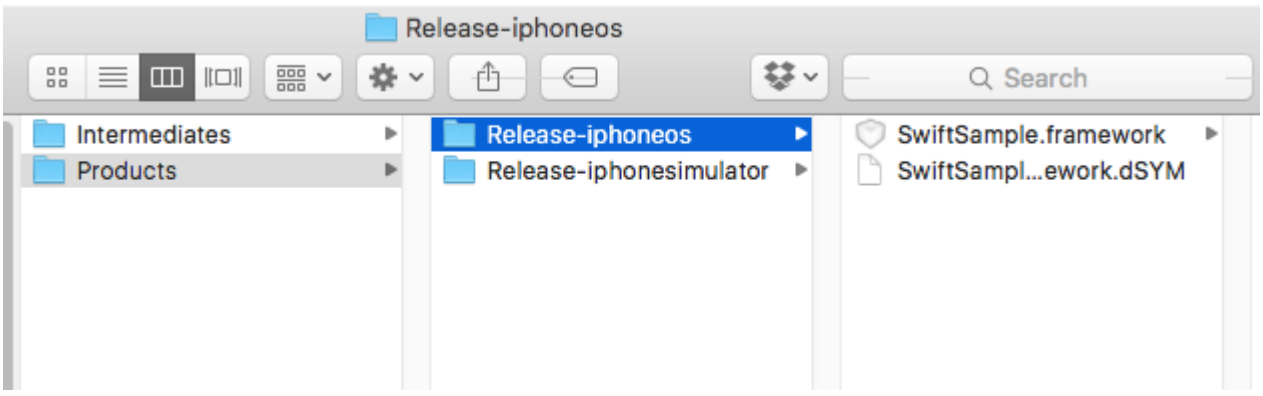

Zur Veröffentlichung für Gerät und Simulator vorgesehen. *


SwiftSample
Generic iOS Device
Build
1 target
Run
Release
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Diagnostics
Build Configuration
Release
Executable
None
Debug executable
Debug Process As
Me (flash)
root
Launch
Automatically
Wait for executable to be launched
Duplicate Scheme
Manage Schemes...
Shared


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot...
Swif...ple
Generic iOS Device
SwiftSample
SwiftSample.h
Info.plist
MyClass.swift
Products
SwiftSample.framework
Basic
All
Build Options
Setting
Build Settings
Running
Testing
Profiling



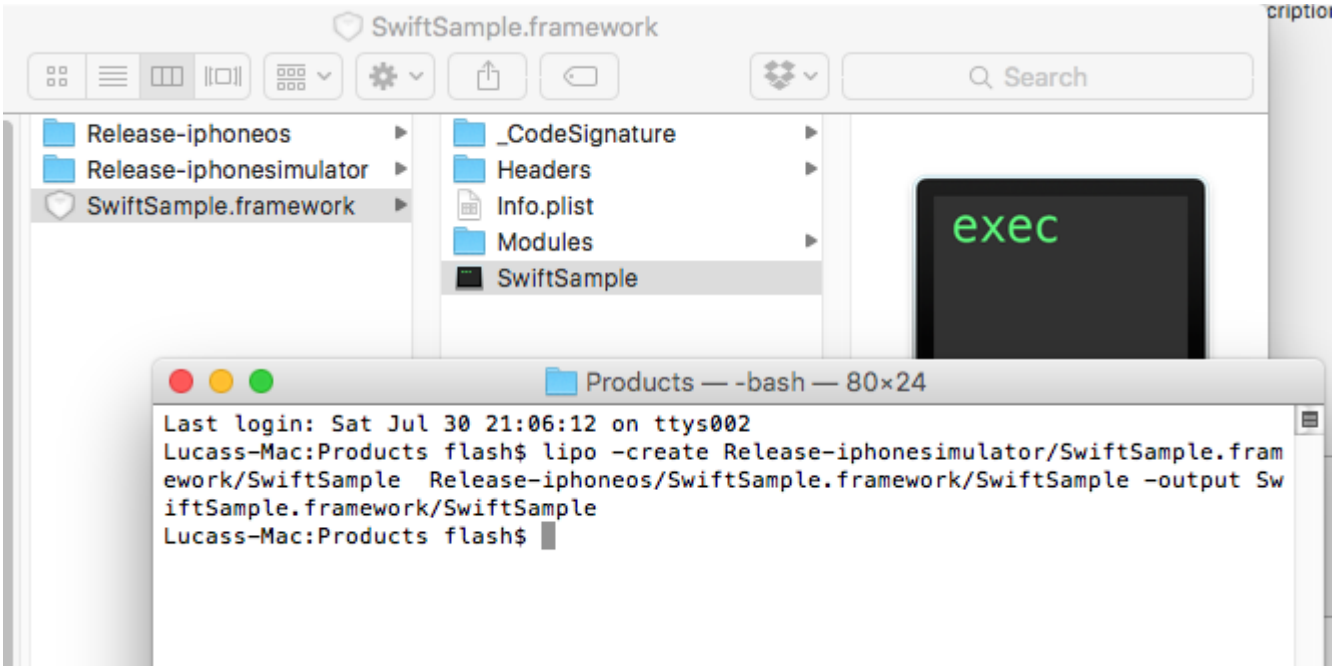



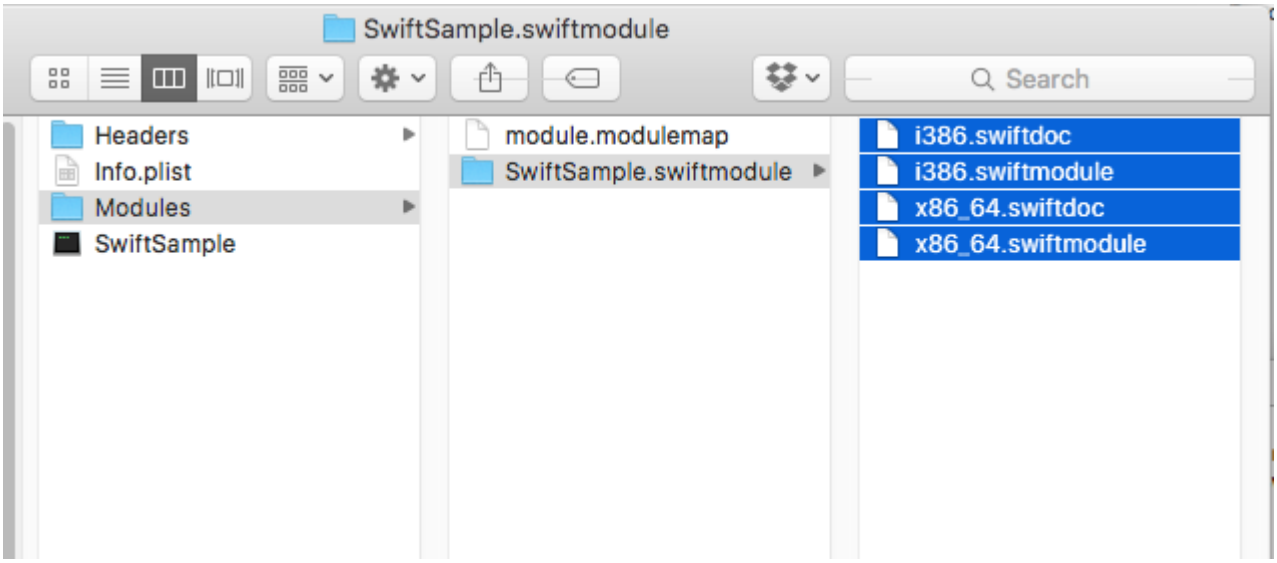

- Nicht nur mit Swift-Bindung verbunden.

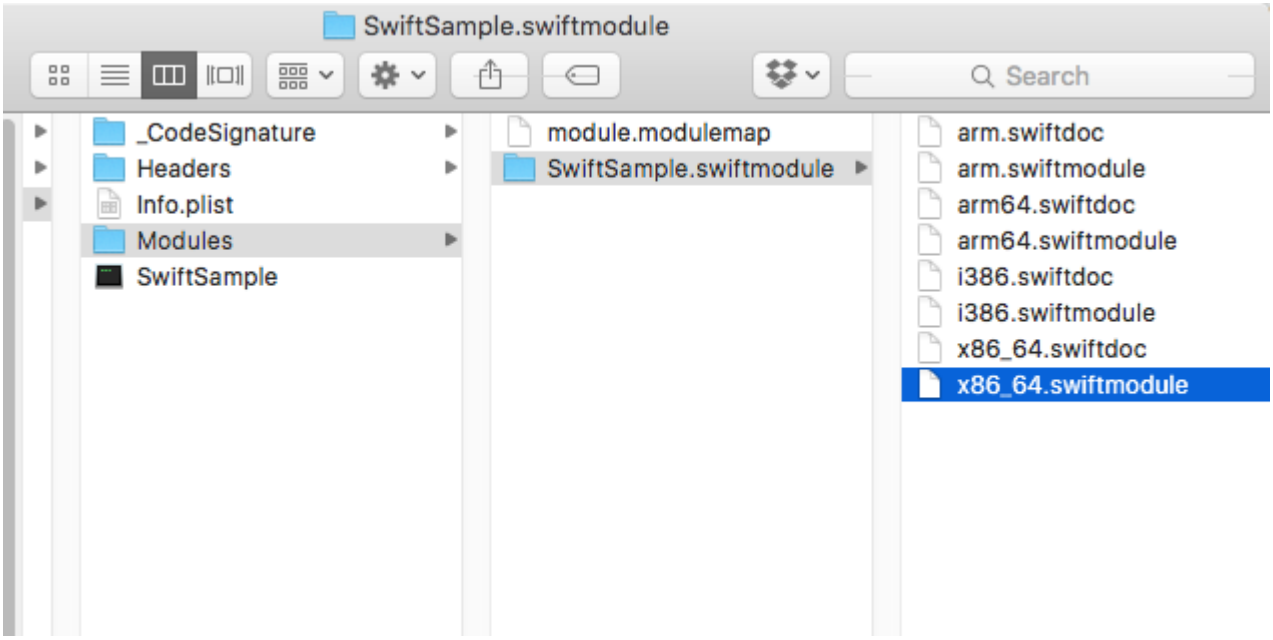
# 2. Erstellen Sie eine Fettbibliothek

Ein Framework enthält mehrere Dateien. Diejenige, die etwas essen muss, ist NAME.framework / NAME (ohne Erweiterung).

- Kopieren Sie Release-iphoneos / NAME.framework in NAME.framework
- Erstellen Sie die FAT-Bibliothek mit:
    - **lipo -create Release-iphonesimulator / NAME.framework / NAME Release-iphoneos / NAME.framework / NAME -out NAME.framework / NAME**
- Kopieren Sie die Dateien unter Release-iphonesimulator / NAME.framework / Modules / NAME.swiftmodule in NAME.framework / Modules / NAME.swiftmodule (bisher enthielten sie nur Dateien aus dem iphoneos).



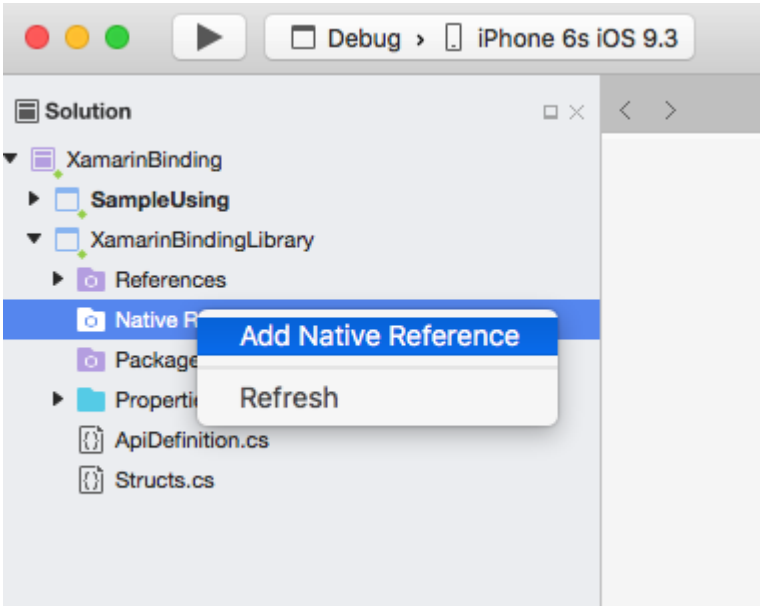



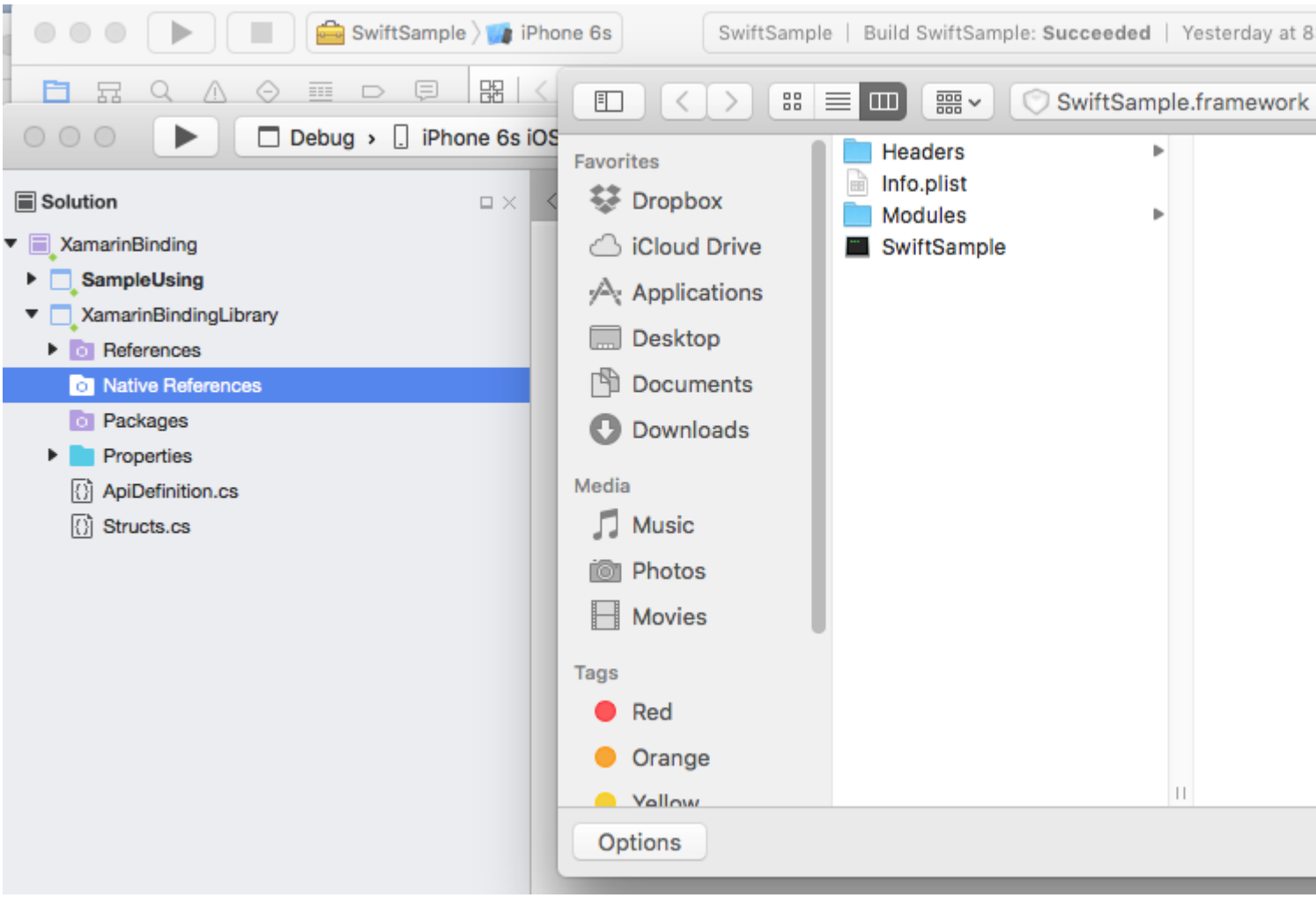


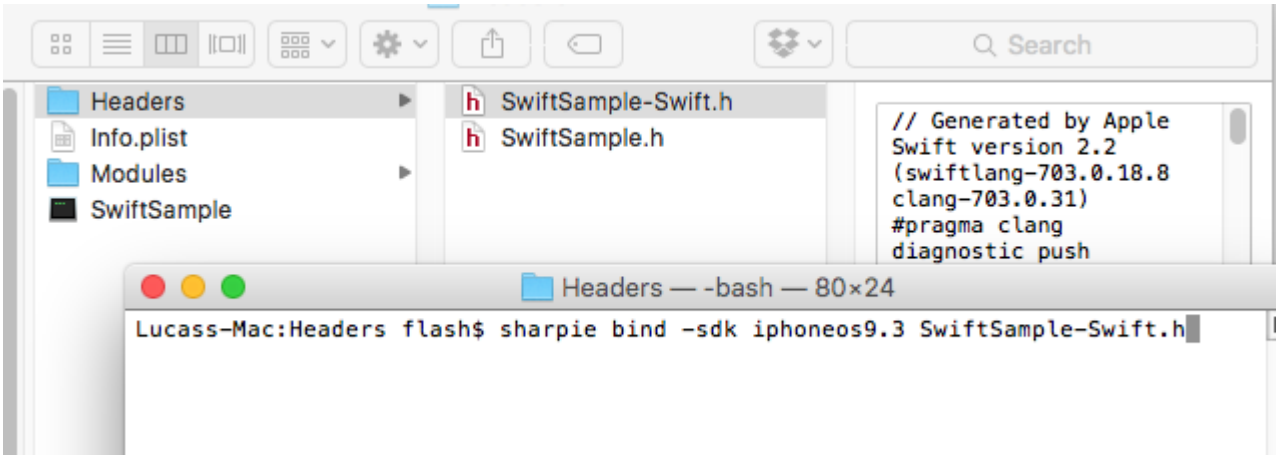

# 3. Importieren Sie die Bibliothek

Ich gehe davon aus, dass Sie das Bindungsprojekt bereits in Datei -> Neu -> iOS -> Bindungsbibliothek erstellt haben.

Xamarin unterstützt den Import von .frameworks. Klicken Sie einfach mit der rechten Maustaste auf "Native References" und klicken Sie auf "Native Reference hinzufügen". Finden Sie das neu erstellte Fat Framework und fügen Sie es hinzu.






## 4. Erstellen Sie die ApiDefinition basierend auf der Datei LIBRARY-Swift.h in den Kopfzeilen.

Sie können es manuell tun, aber es ist nicht schön. Sie können Objetive Sharpie verwenden. Das Tool, mit dem Xamarin seine eigenen Bibliotheken bindet.

Verwendung unter https://developer.xamarin.com/guides/cross-platform/macios/binding/objective-sharpie/

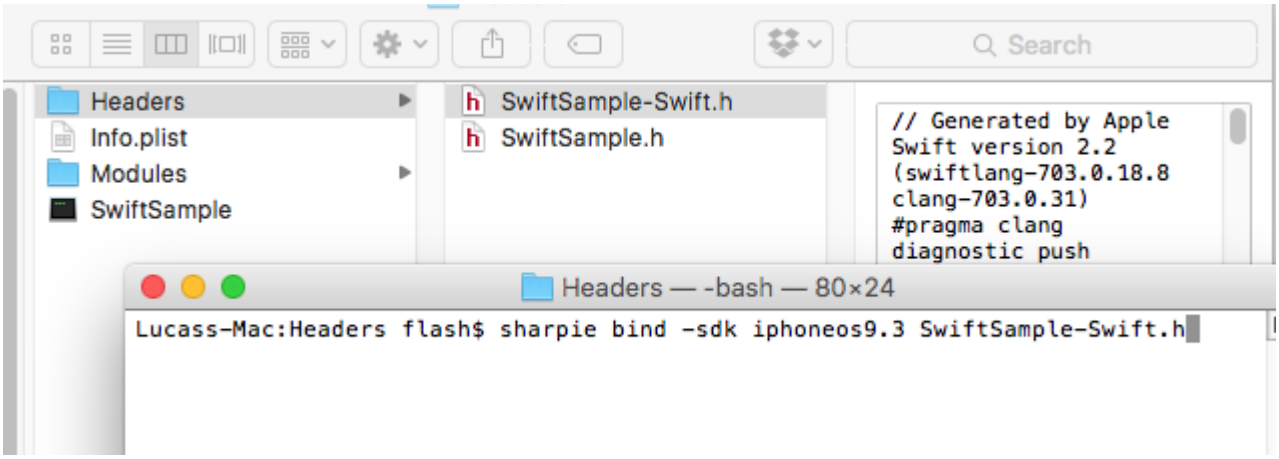
Der ***Basisbefehl lautet*** etwa: ***sharpie bind -sdk iphoneos9.3 NAME-Swift.h***

Wenn Sie eine `System.Reflection.TargetInvocationException` , liegt dies wahrscheinlich daran, dass Sie eine andere SDK-Version installiert haben. Führen Sie den folgenden Befehl aus, um das installierte iPhone OS SDK zu überprüfen:

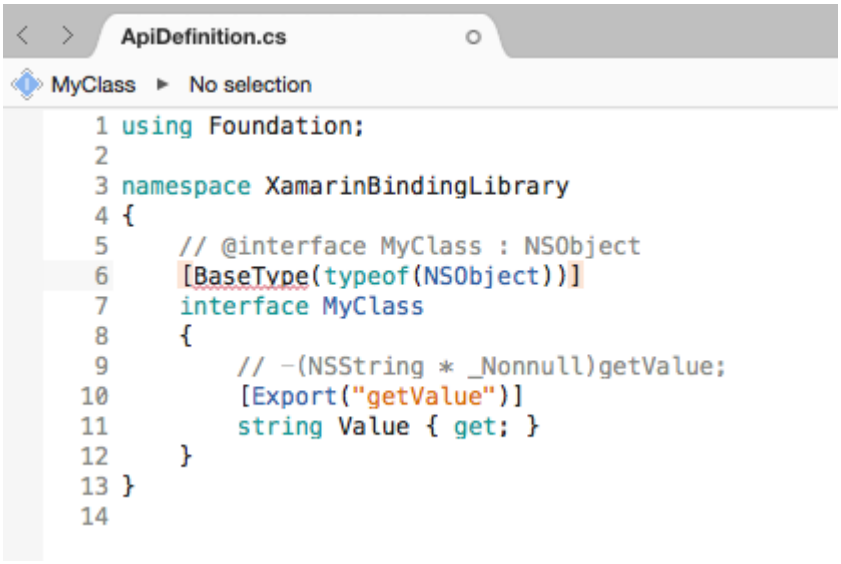
```
sharpie xcode -sdks
```

Die Datei ***NAME-Swift.h*** befindet sich in ***NAME.framework / Headers / NAME-Swift.h***

Hinweis: Die schnellen Klassen müssen von "NSObject" erben. Andernfalls importiert ***NAME-Swift.h*** Ihre Klassen nicht und Objetive Sharpie konvertiert nichts.




Ersetzen Sie den Inhalt Ihres Bindungsprojekts ApiDefinition.cs durch das neu erstellte.




# 5. Ändern Sie alle [Protocol] und [BaseType], um den Klassennamen in die Objective-C-

# Laufzeit aufzunehmen.

Wenn die ursprüngliche Swift-Klasse oder das ursprüngliche Protokoll nicht die @objc-Annotation (MyClass) wie in Schritt 1.1 angegeben enthält, werden die internen Objective-C-Namen geändert. Sie müssen also die richtige Zuordnung vornehmen.

Alle Namen sind in der Datei NAME-Swift.h im folgenden Format verfügbar:

```
SWIFT_CLASS("_TtC11SwiftSample7MyClass")
@interface MyClass : NSObject
```

Und

```
SWIFT_PROTOCOL("_TtP6Charts17ChartDataProvider_")
@protocol ChartDataProvider
```

Um den Namen festzulegen, verwenden Sie BaseTypeAttribute.Name https://developer.xamarin.com/guides/cross-platform/macios/binding/binding-types-reference/#BaseType.Name-Eigenschaft für Klassen und ProcotolAttribute.Name https: // developer.xamarin.com/api/property/MonoTouch.Foundation.ProtocolAttribute.Name/ für Protokolle.

```
[BaseType(typeof(NSObject), Name = "_TtC11SwiftSample7MyClass")]
interface MyClass
```

Manuell zu tun ist nicht cool. Sie können dieses Tool https://github.com/Flash3001/SwiftClassify verwenden , um alle Namen einzufügen. (Es wird gerade gearbeitet. Aber es ist ziemlich einfach, wenn Sie sich nur den Code ansehen, erfahren Sie, wie er funktioniert.)

# 6.1 Alle Swift-Abhängigkeiten einschließen.

Wenn Sie versuchen, die Bibliothek in einer App zu verbrauchen und sie sofort auszuführen, stürzt sie ab. Der Fehler ist auf das Fehlen von libswiftCore.dylib zurückzuführen

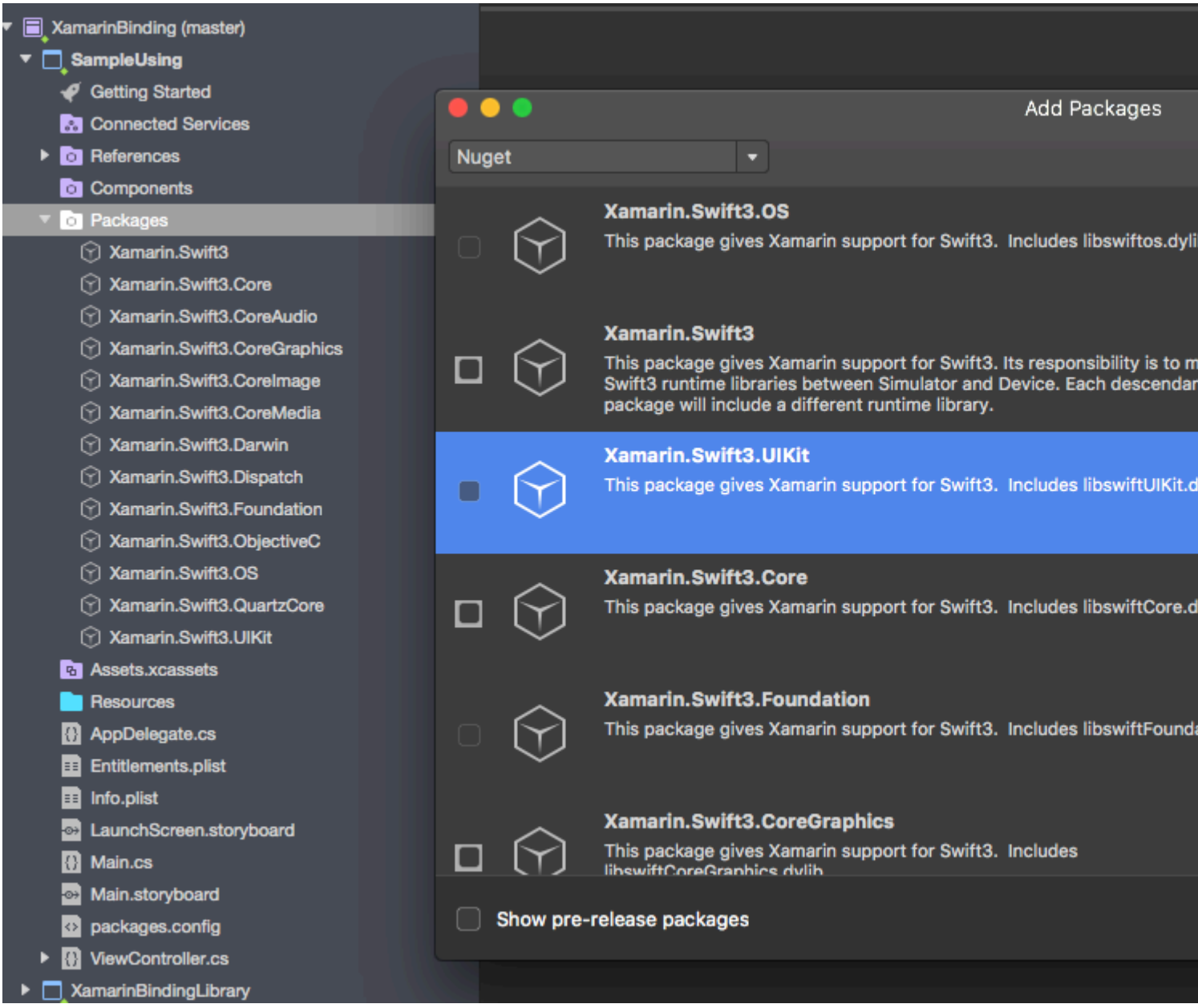
Etwas wie das:

```
Dyld Error Message:
  Library not loaded: @rpath/libswiftCore.dylib
  Referenced from: /Users/USER/Library/Developer/CoreSimulator/Devices/AC440891-C819-4050-
8CAB-CE15AB4B3830/data/Containers/Bundle/Application/27D2EC87-5042-4FA7-9B80-
A24A8971FB48/SampleUsing.app/Frameworks/SwiftSample.framework/SwiftSample
  Reason: image not found
```

Xamarin.iOS unterstützt die Bindung einer Swift-Bibliothek nicht offiziell. Daher müssen Sie die schnellen Kernbibliotheken manuell in die Ordner Frameworks und SwiftSupport aufnehmen. Die Dateien für den Ordner "Frameworks" unterscheiden sich für Simulator und Gerät. Sie können in

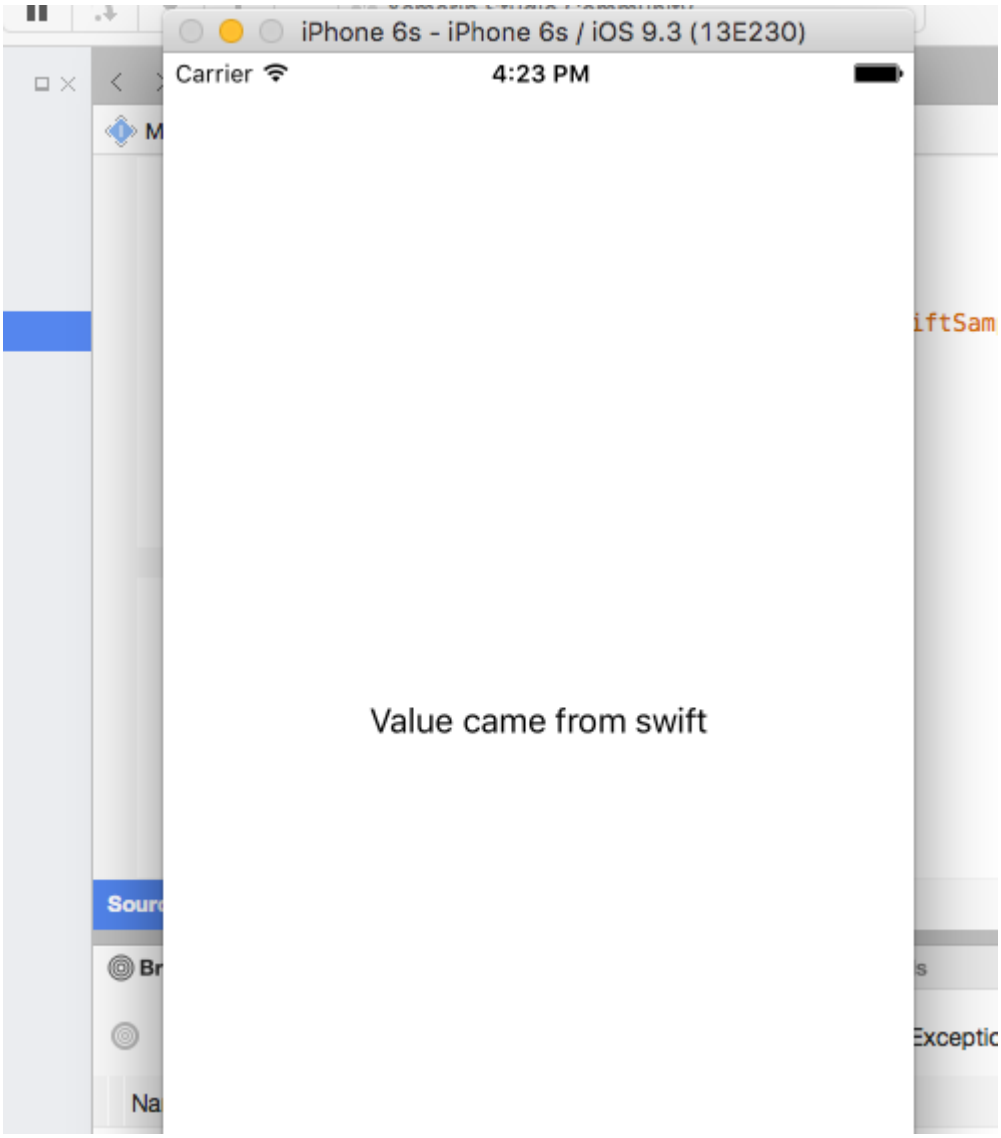
/Applications/Xcode.app/Contents/Developer//XcodeDefault.xctoolchain/usr/lib/swift.Toolchains gefunden werden

Anstatt die Dateien innerhalb des Framework-Ordners manuell zu kopieren, können Sie die Bibliothek https://github.com/Flash3001/Xamarin.Swift3.Support verwenden . Es enthält alle Abhängigkeiten, die Swift 3.1 benötigt, und jede in einem einzigen NuGet-Paket.




Wie Sie sehen, ist das Nuget-Paket in der Consumer-App enthalten, nicht in der Bindung selbst. Wenn Sie versuchen, es in die Bindung einzubinden, werden Kompilierungsfehler angezeigt.

Wenn Sie ein Nuget-Paket erstellen, können Sie Nuget anweisen, es als Abhängigkeit hinzuzufügen.



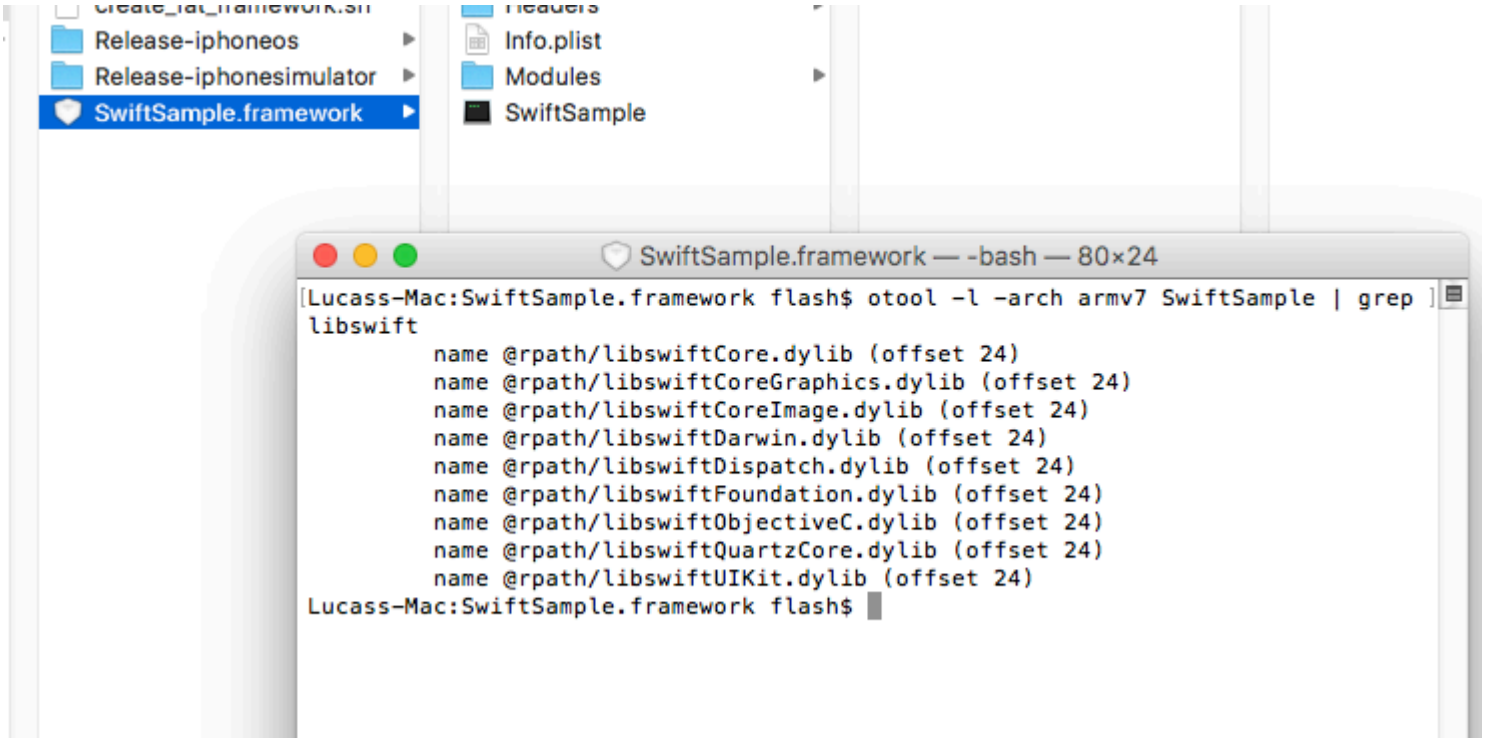

## 6.2. Herausfinden, welche Swift-Abhängigkeiten enthalten sind.

Es ist wichtig, jedes Paket herauszufinden, das Sie in Ihr Projekt aufnehmen müssen. Eine einfache Bindung benötigt normalerweise:

```
    libswiftCore.dylib
    libswiftCoreGraphics.dylib
    libswiftCoreImage.dylib
    libswiftDarwin.dylib
    libswiftDispatch.dylib
    libswiftFoundation.dylib
    libswiftObjectiveC.dylib
    libswiftQuartzCore.dylib
    libswiftUIKit.dylib
```

Um jede Abhängigkeit aufzulisten, können Sie den folgenden Befehl in Ihrem LibraryName.framework ausführen

```
otool -l -arch armv7 LibraryName | grep libswift
```



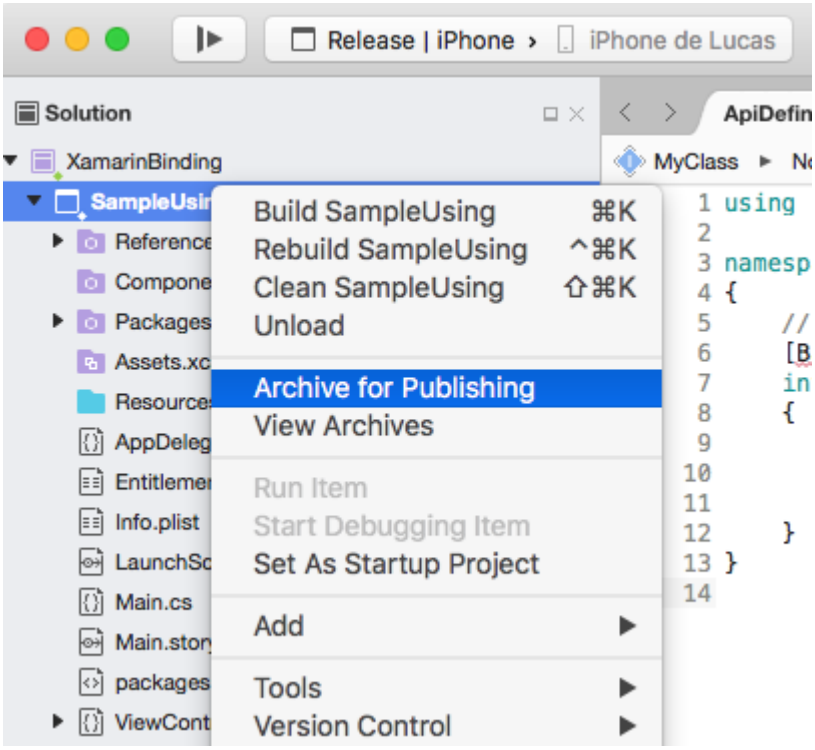

Schließen Sie nicht jedes Paket ein, das in NuGet für Swift3 verfügbar ist, da dies Ihre App-Größe erhöhen könnte.

## 7. Fügen Sie SwiftSupport hinzu, um die App in den AppStore zu verschieben.

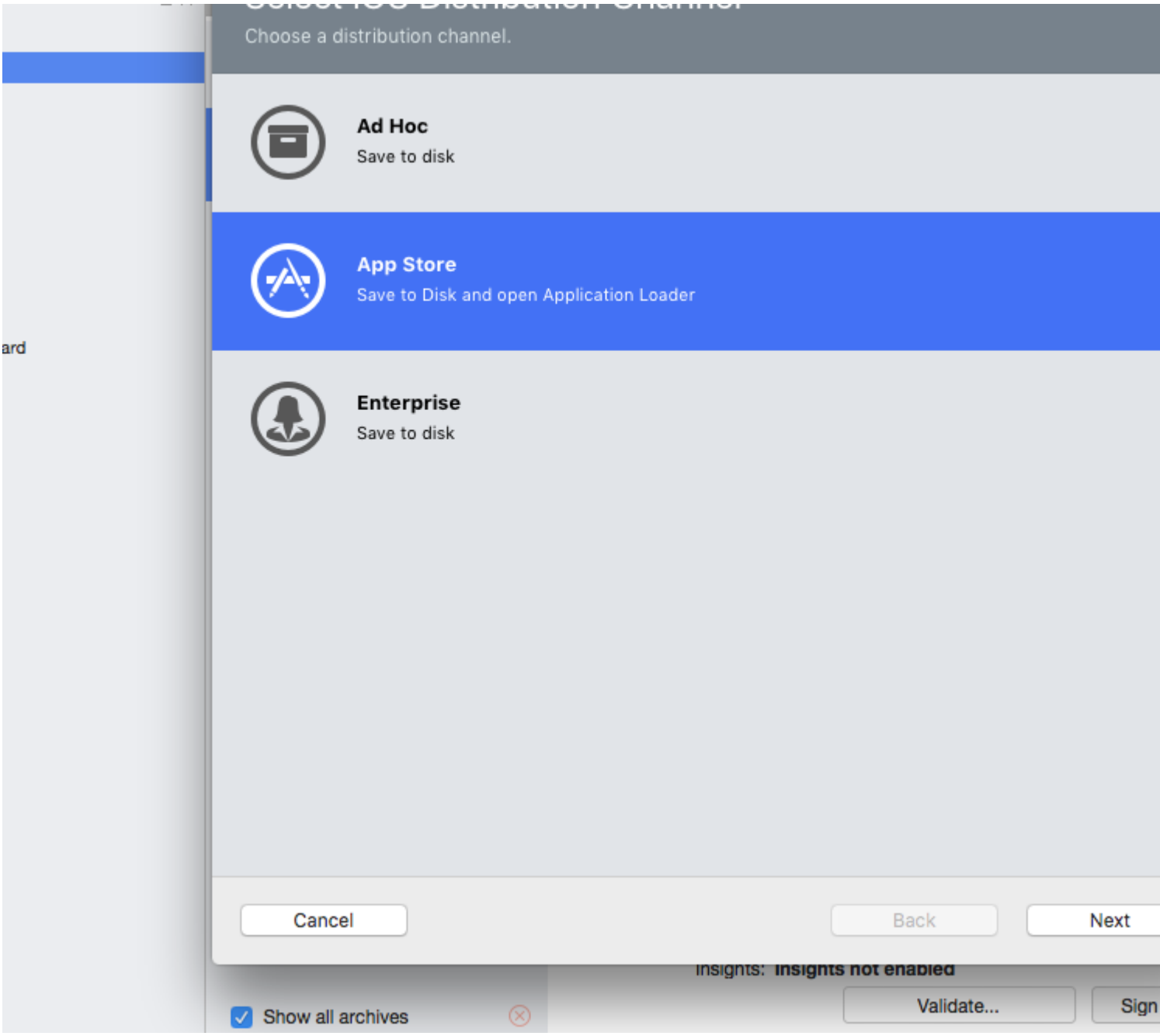
Apple verlangt, dass Ihre App zusammen mit Ihrem Payload-Ordner mit einem SwiftSupport-Ordner gesendet wird. Beide befinden sich in Ihrem IPA-Paket.

Sie können dieses Skript https://github.com/bq/ipa-packager verwenden , um diese Arbeit für Sie auszuführen.

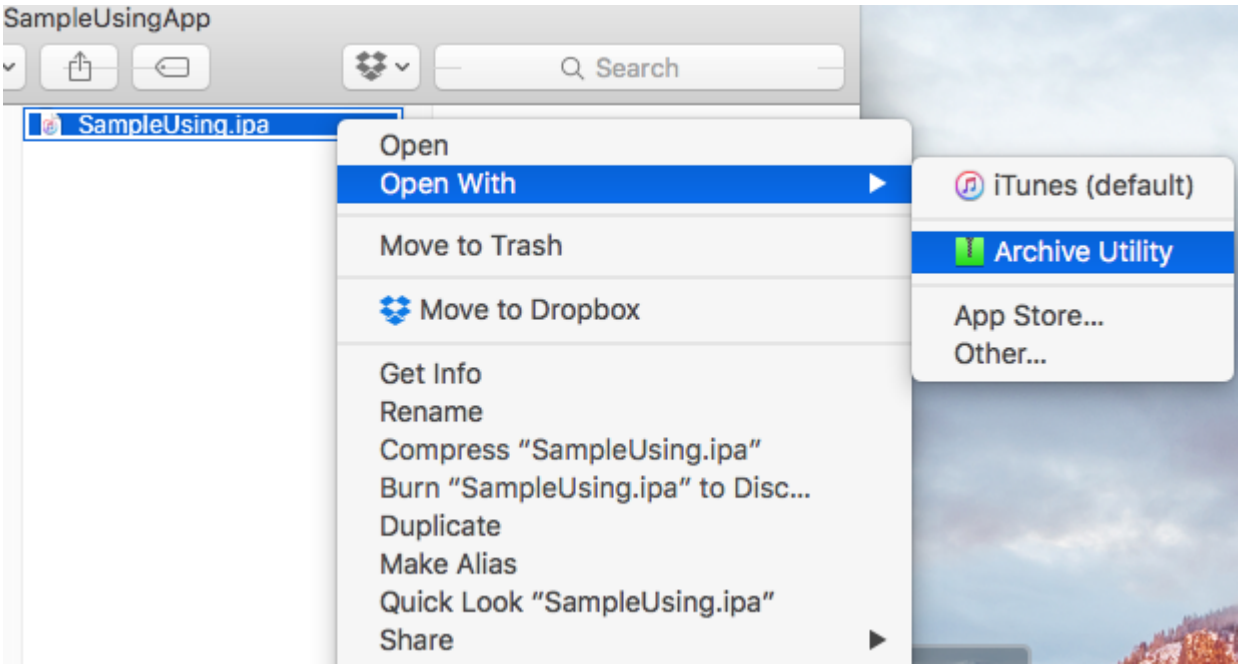
Dieser Prozess ist der einzige, den der Konsument der Bibliothek manuell durchführen muss. Jedes Mal, wenn er versucht, die App in den AppStore zu verschieben.



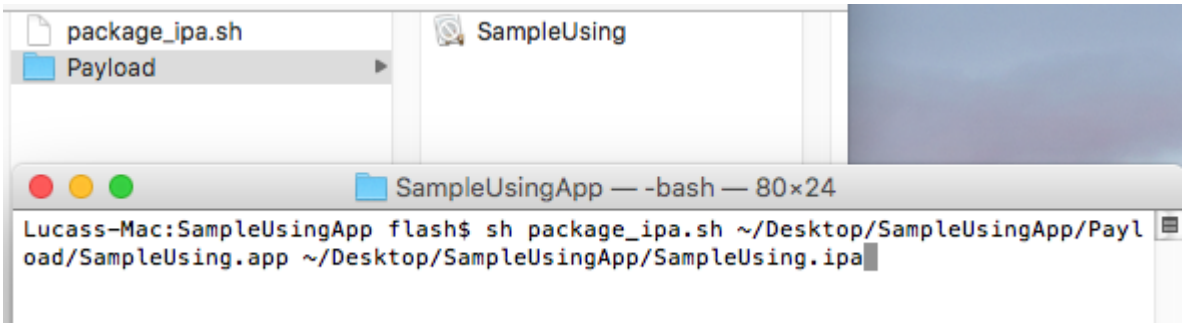

Klicken Sie auf "Signieren und Verteilen" und auf Festplatte speichern



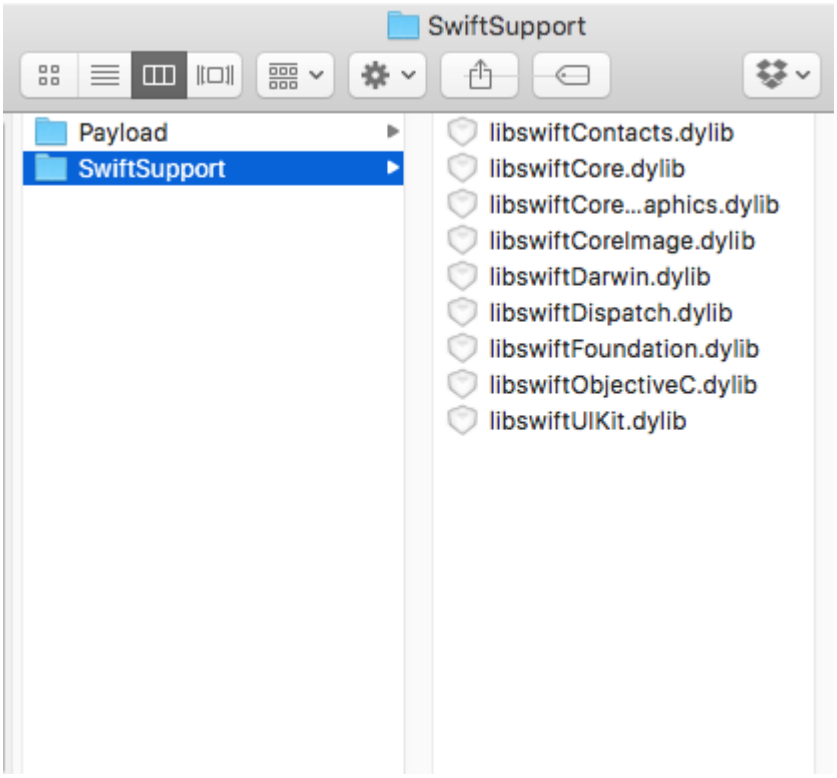

Entpacken Sie Ihre .IPA




Erstellen Sie den neuen IPA mit dem zuvor genannten Skript




Wenn Sie die Datei entpacken, wird sie den SwiftSupport-Ordner enthalten.



# Bemerkungen

Wenn Sie eine Bibliothek in Xcode erstellen, haben Sie die Möglichkeit, die schnellen Bibliotheken einzubinden. Nicht! Sie werden in Ihre endgültige App als NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib aufgenommen, müssen jedoch als NAME.app/Frameworks/libswift*.dylib enthalten sein

Sie finden diese Informationen an anderer Stelle, aber es ist erwähnenswert: Fügen Sie Bitcode nicht in die Bibliothek ein. Im Moment enthält Xamarin Bitcode für iOS nicht und Apple erfordert, dass alle Bibliotheken die gleichen Architekturen unterstützen.

# Haftungsausschluss

Dieses Handbuch wurde ursprünglich von Lucas Teixeira erstellt . Alle Credits gehören ihm. Vielen Dank, Lucas.

Binden Sie schnelle Bibliotheken online lesen: https://riptutorial.com/de/xamarin-ios/topic/6091/binden-sie-schnelle-bibliotheken

# Kapitel 10: Erstellen und Verwenden von benutzerdefinierten Prototyp-Tabellenzellen in xamarin.iOS mithilfe des Storyboards

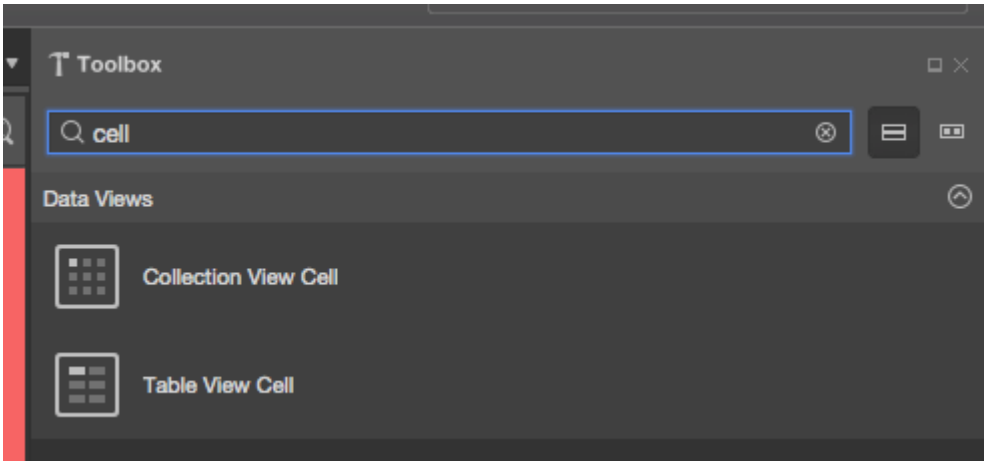
## Examples

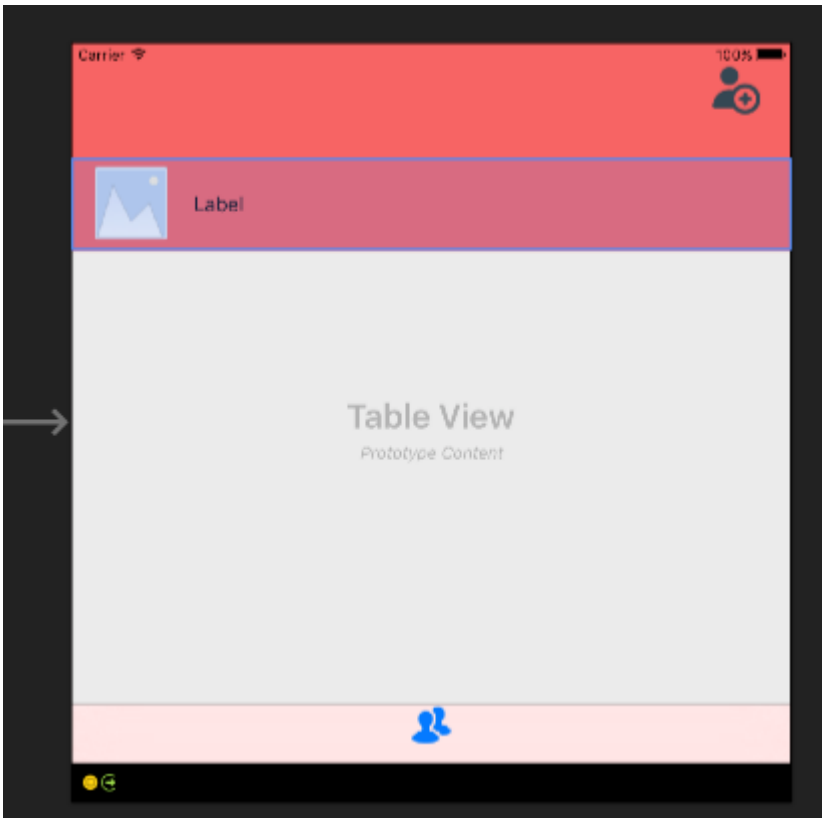
### Erstellen Sie eine benutzerdefinierte Zelle mithilfe des Storyboards

Öffnen Sie das Storyboard, wo Sie Ihren ViewController mit TableView haben:

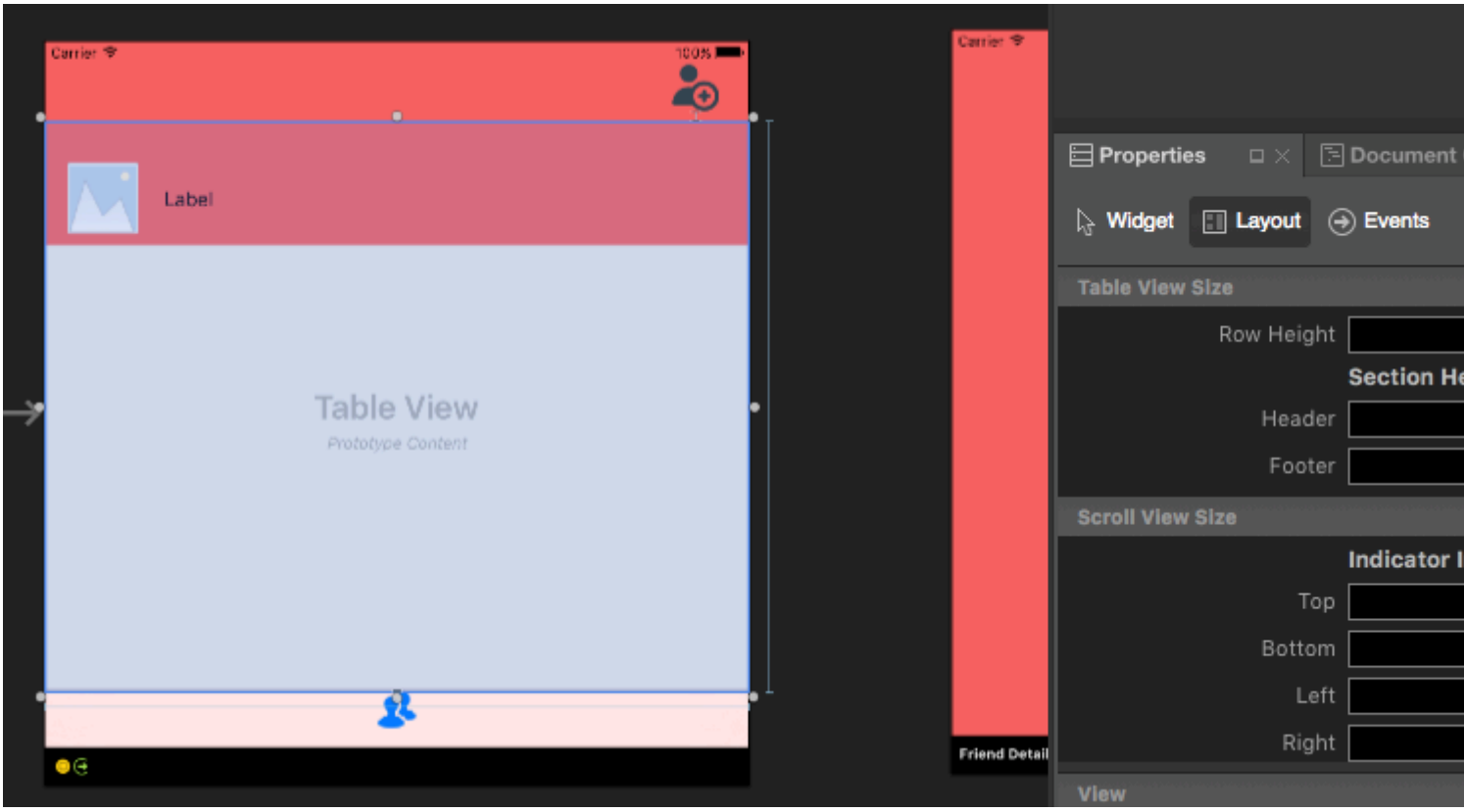
Prototypzelle hinzufügen (falls noch keine Zelle hinzugefügt wurde):

Anpassen der Zelle nach Belieben (in meinem Fall gibt es benutzerdefinierte UIImage und Label):



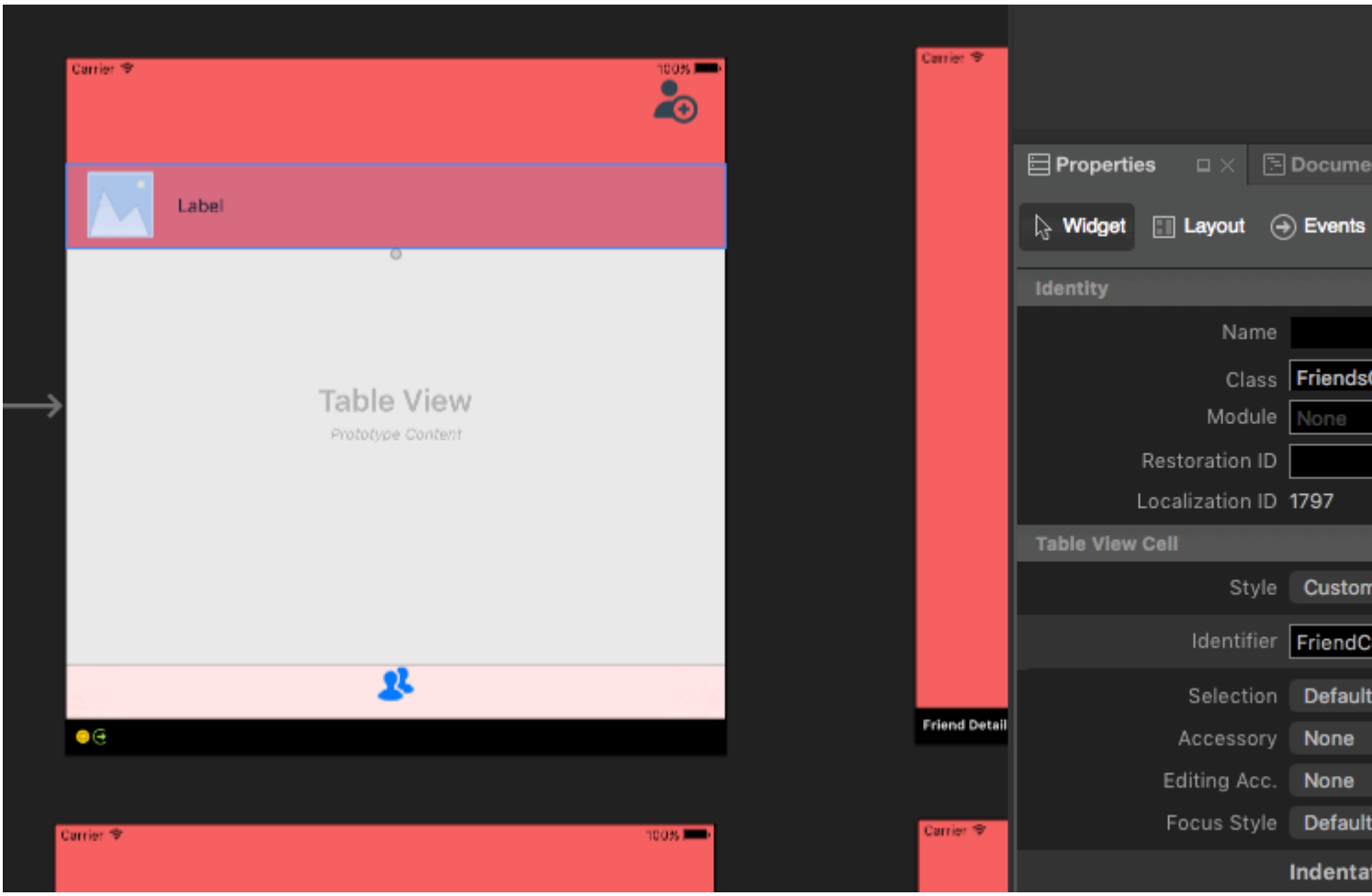


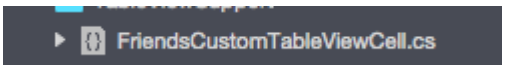

Denken Sie daran, die Höhe der Zelle einzustellen. Wählen Sie dazu Ihre gesamte TableView aus und wählen Sie im Eigenschaftenfenster die Registerkarte "Layout". Oben im Eigenschaftenfenster sollte "Zeilenhöhe" angezeigt werden - geben Sie den entsprechenden Wert ein:




Wählen Sie nun erneut die Prototypzelle aus. Geben Sie im Eigenschaftenfenster den Namen der Klasse ein (die Code-Behind-Klasse wird erstellt). In meinem Fall ist dies

"FriendsCustomTableViewCell". Danach geben Sie "Identifier" für Ihre Zelle an. Wie Sie sehen können, ist meine "FriendCell". Als letztes zu setzendes Element wird die Eigenschaft "Style" auf custom gesetzt. Das Feld "Name" sollte leer sein. Wenn Sie nach der Eingabe von "Class" auf "Eingabe" klicken, wird automatisch eine Code-Behind-Datei erstellt:




Code hinter der Zelle sollte jetzt wie folgt aussehen:

```
public partial class FriendsCustomTableViewCell : UITableViewCell
{
    public FriendsCustomTableViewCell (IntPtr handle) : base (handle)
    {
    }

    public FriendsCustomTableViewCell(NSString cellId, string friendName, UIImage friendPhoto)
: base (UITableViewCellStyle.Default, cellId)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
    }

    //This methods is to update cell data when reuse:
    public void UpdateCellData(string friendName, UIImage friendPhoto)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
```

```
    }
}
```

In UITableViewSource müssen Sie cellIdentifier an der Spitze der Klasse deklarieren (in meinem Fall "FriendCell"), und in der "GetCell" -Methode müssen Sie Zellen umwandeln und Daten dafür festlegen:

```
string cellIdentifier = "FriendCell";

public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
{
    FriendsCustomTableViewCell cell = (FriendsCustomTableViewCell)
tableView.DequeueReusableCell(cellIdentifier);
    Friend friend = _friends[indexPath.Row];

    //---- if there are no cells to reuse, create a new one
    if (cell == null)
    { cell = new FriendsCustomTableViewCell(new NSString(cellIdentifier), friend.FriendName,
new UIImage(NSData.FromArray(friend.FriendPhoto))); }

    cell.UpdateCellData(friend.UserName, new UIImage(NSData.FromArray(friend.FriendPhoto)));

    return cell;
}
```

Erstellen und Verwenden von benutzerdefinierten Prototyp-Tabellenzellen in xamarin.iOS mithilfe des Storyboards online lesen: https://riptutorial.com/de/xamarin-ios/topic/5907/erstellen-und-verwenden-von-benutzerdefinierten-prototyp-tabellenzellen-in-xamarin-ios-mithilfe-des-storyboards

# Kapitel 11: Fügen Sie PullToRefresh zu UITableView hinzu

## Bemerkungen

Objektreferenzen:

UITableView-Tabelle;
TableSource tableSource;
bool useRefreshControl = false;
UIRefreshControl RefreshControl;
Listet tableItems auf;

TableSource und TableItem sind benutzerdefinierte Klassen

Für das vollständige Beispiel können Sie sich anmelden:
https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/PullToRefresh

## Examples

### Hinzufügen von UIRefreshControl zu UITableView

```
public override async void ViewDidLoad(){
base.ViewDidLoad();
// Perform any additional setup after loading the view, typically from a nib.

Title = "Pull to Refresh Sample";
table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
//table.AutoresizingMask = UIViewAutoresizing.All;
tableItems = new List<TableItem>();
tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
tableSource = new TableSource(tableItems);
table.Source = tableSource;

await RefreshAsync();

AddRefreshControl();

Add(table);
table.Add(RefreshControl);
}

async Task RefreshAsync()
{
// only activate the refresh control if the feature is available
if (useRefreshControl)
```

```
    RefreshControl.BeginRefreshing();

if (useRefreshControl)
    RefreshControl.EndRefreshing();

    table.ReloadData();
}

#region * iOS Specific Code
// This method will add the UIRefreshControl to the table view if
// it is available, ie, we are running on iOS 6+
void AddRefreshControl()
{
if (UIDevice.CurrentDevice.CheckSystemVersion(6, 0))
{
    // the refresh control is available, let's add it
    RefreshControl = new UIRefreshControl();
    RefreshControl.ValueChanged += async (sender, e) =>
    {
        tableItems.Add(new TableItem("Bulbs") { ImageName = "Bulbs.jpg" });
        await RefreshAsync();
    };
    useRefreshControl = true;
   }
}
#endregion
```

Fügen Sie PullToRefresh zu UITableView hinzu online lesen: https://riptutorial.com/de/xamarin-ios/topic/6565/fugen-sie-pulltorefresh-zu-uitableview-hinzu

# Kapitel 12: Gleichzeitiges Programmieren in Xamarin.iOS

## Examples

### Manipulieren der Benutzeroberfläche von Hintergrundthreads

Hintergrund-Threads können die Benutzeroberfläche nicht ändern. Fast alle UIKit-Methoden müssen im Hauptthread aufgerufen werden.

Aus einer Unterklasse von `NSObject` (einschließlich `UIViewController` oder `UIView` ):

```
InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

Aus einer Standardklasse C #:

```
UIApplication.SharedApplication.InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

`InvokeOnMainThread` wartet, bis der Code im Hauptthread ausgeführt wird, bevor Sie fortfahren. Wenn Sie nicht warten müssen, verwenden Sie `BeginInvokeOnMainThread` .

### Async verwenden und warten

Sie können asynchrone Methoden verwenden, um asynchrone Ausführungen auszuführen. Zum Beispiel POST- und GET-Anfragen. Nehmen wir an, Sie erhalten Ihre Datenmethode unten.

```
Task<List> GetDataFromServer(int type);
```

Sie können diese Methode wie unten gezeigt aufrufen

```
var result = await GetDataFromServer(1);
```

In der Praxis wird diese Methode jedoch in einer Service-Layer-Schnittstelle sein. Am besten erstellen Sie hierfür eine separate Methode, um diese aufzurufen und die unten gezeigte Benutzeroberfläche zu aktualisieren.

```
//Calling from viewDidLoad
void async ViewDidLoad()
{
    await GetDataListFromServer(1);
```

```
    //Do Something else
}

//New method call to handle the async task
private async Task GetArchivedListFromServer(int type)
{
    var result = await GetDataFromServer(type);
    DataList.AddRange(result.toList());
    tableView.ReloadData();
}
```

Im obigen Code-Snippet wird die GetDataListFromServer-Methode aufgerufen und die Webanforderung wird gesendet. Trotzdem wird der UI-Thread nicht blockiert, bis er die Antwort vom Server erhält. Nach dem `await GetDataListFromServer(1)` wird die Zeile nach unten `await GetDataListFromServer(1)` . Innerhalb der `private async Task GetArchivedListFromServer(int type)` - Methode wartet sie jedoch, bis sie die Antwort vom Server erhält, um die Zeilen nach `var result = await GetDataFromServer(type);` auszuführen `var result = await GetDataFromServer(type);` .

Gleichzeitiges Programmieren in Xamarin.iOS online lesen: https://riptutorial.com/de/xamarin-ios/topic/1364/gleichzeitiges-programmieren-in-xamarin-ios

# Kapitel 13: Hinzufügen von UIRefreshControl zu einer Tabellensicht

## Examples

### Hinzufügen eines UIRefreshControl zu einer TableView

Annahmen:

TableView - Verweis auf die TableView

DataSource - ist eine Klasse, die UITableViewSource erbt

DataSource.Objects - ist eine öffentliche Liste <object> (), auf die der UIViewController zugreifen kann

```
private UIRefreshControl refreshControl;

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

    // Set the DataSource for the TableView
    TableView.Source = dataSource = new DataSource(this);

    // Create the UIRefreshControl
    refreshControl = new UIRefreshControl();

    // Handle the pullDownToRefresh event
    refreshControl.ValueChanged += refreshTable;

    // Add the UIRefreshControl to the TableView
    TableView.AddSubview(refreshControl);
}

private void refreshTable(object sender, EventArgs e)
{
    fetchData();
    refreshControl.EndRefreshing();
    TableView.ReloadData();
}

private void fetchData()
{
    var objects = new List<object>();
    // fetch data and store in objects.
    dataSource.Objects = objects;
}
```

Hinzufügen von UIRefreshControl zu einer Tabellensicht online lesen: https://riptutorial.com/de/xamarin-ios/topic/4642/hinzufugen-von-uirefreshcontrol-zu-einer-tabellensicht

# Kapitel 14: Hinzufügen von UIRefreshControl zu einer Tabellensicht

## Examples

### Fügen Sie ein einfaches UIRefreshControl zu einer UIScrollView hinzu

Wir gehen von einer voll funktionsfähigen `UIScrollview` namens `_scrollView` .

Beachten Sie, dass `UITableView` und `UICollectionView` auch Bildlaufansichten sind. Daher könnten die folgenden Beispiele für diese `UITableView` `UICollectionView` .

Erstens, Erstellung und Zuordnung

```
UIRefreshControl refreshControl = new UIRefreshControl();
```

Zweitens, Verbinden des Aktualisierungsereignisses mit einer Methode. Dafür gibt es verschiedene Möglichkeiten.

# Stil 1:

```
refreshControl.ValueChanged += (object sender, EventArgs e) => MyMethodCall();
```

# Stil 2:

```
refreshControl.ValueChanged += (object sender, EventArgs e) =>
{
    //Write code here
};
```

# Stil 3:

```
refreshControl.ValueChanged += HandleRefreshValueChanged;

void HandleRefreshValueChanged(object sender, EventArgs e)
{
    //Write code here
}
```

Als drittes und letztes fügen Sie die Aktualisierungssteuerung selbst zu unserer Bildlaufansicht hinzu.

```
_scrollView.AddSubview(refreshControl);
```

Hinzufügen von UIRefreshControl zu einer Tabellensicht online lesen: https://riptutorial.com/de/xamarin-ios/topic/8371/hinzufugen-von-uirefreshcontrol-zu-einer-tabellensicht

# Kapitel 15: Methoden zur Größenänderung für UIImage

## Examples

### Bildgröße ändern - mit Seitenverhältnis

```
// resize the image to be contained within a maximum width and height, keeping aspect ratio
public static UIImage MaxResizeImage(this UIImage sourceImage, float maxWidth, float
maxHeight)
{
    var sourceSize = sourceImage.Size;
    var maxResizeFactor = Math.Min(maxWidth / sourceSize.Width, maxHeight /
sourceSize.Height);
    if (maxResizeFactor > 1) return sourceImage;
    var width = maxResizeFactor * sourceSize.Width;
    var height = maxResizeFactor * sourceSize.Height;
    UIGraphics.BeginImageContext(new CGSize(width, height));
    sourceImage.Draw(new CGRect(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

### Bildgröße ändern - ohne Seitenverhältnis

```
// resize the image (without trying to maintain aspect ratio)
public static UIImage ResizeImage(this UIImage sourceImage, float width, float height)
{
    UIGraphics.BeginImageContext(new SizeF(width, height));
    sourceImage.Draw(new RectangleF(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

### Bild zuschneiden ohne Größe ändern

```
// crop the image, without resizing
public static UIImage CropImage(this UIImage sourceImage, int crop_x, int crop_y, int width,
int height)
{
    var imgSize = sourceImage.Size;
    UIGraphics.BeginImageContext(new SizeF(width, height));
    var context = UIGraphics.GetCurrentContext();
    var clippedRect = new RectangleF(0, 0, width, height);
    context.ClipToRect(clippedRect);
    var drawRect = new CGRect(-crop_x, -crop_y, imgSize.Width, imgSize.Height);
    sourceImage.Draw(drawRect);
    var modifiedImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
```

```
    return modifiedImage;
}
```

Methoden zur Größenänderung für UIImage online lesen: https://riptutorial.com/de/xamarin-ios/topic/6542/methoden-zur-gro-enanderung-fur-uiimage

# Kapitel 16: So verwenden Sie Asset-Asset-Kataloge

## Examples

### Asset-Kataloge verwenden

Um einen Asset-Katalog zu verwenden, müssen Sie Folgendes tun:

1. Doppelklicken Sie im Projektmappen-Explorer auf die Datei Info.plist, um sie zur Bearbeitung zu öffnen.
2. Scrollen Sie nach unten zum Abschnitt App-Symbole.
3. Stellen Sie sicher, dass in der Dropdown-Liste Quelle AppIcons ausgewählt ist.
4. Doppelklicken Sie im Projektmappen-Explorer auf die Datei Assets.xcassets, um sie zur Bearbeitung zu öffnen.
5. Wählen Sie AppIcons aus der Liste der Assets aus, um den Symbol-Editor anzuzeigen.
6. Klicken Sie entweder auf den angegebenen Symboltyp und wählen Sie eine Bilddatei für den gewünschten Typ / die gewünschte Größe aus, oder ziehen Sie ein Bild aus einem Ordner und legen Sie es auf die gewünschte Größe.
7. Klicken Sie auf die Schaltfläche Öffnen, um das Bild in das Projekt aufzunehmen, und legen Sie es im Xcasset fest.

So verwenden Sie Asset-Asset-Kataloge online lesen: https://riptutorial.com/de/xamarin-ios/topic/6539/so-verwenden-sie-asset-asset-kataloge

# Kapitel 17: Steuern des Screenshots im iOS Multitasking Switcher

## Einführung

Im App-Programmierhandbuch für iOS :

Entfernen Sie vertrauliche Informationen aus Ansichten, bevor Sie in den Hintergrund wechseln.

Wenn eine App in den Hintergrund wechselt, erstellt das System eine Momentaufnahme des Hauptfensters der App, die dann beim Übergang der App in den Vordergrund kurz angezeigt wird.

## Bemerkungen

Angepasst an die eigentliche StackOverflow-Frage Steuern des Screenshots im iOS7-Multitasking-Switcher und Antwort auf Obj-c Answer

## Examples

### Zeigen Sie ein Bild für den Schnappschuss

```
public override void DidEnterBackground(UIApplication application)
 {
    //to add the background image in place of 'active' image
    var backgroundImage = new UIImageView();
    backgroundImage.Tag = 1234;
    backgroundImage.Image = UIImage.FromBundle("Background");
    backgroundImage.Frame = this.window.Frame;
    this.window.AddSubview(backgroundImage);
    this.window.BringSubviewToFront(backgroundImage);
}

public override void WillEnterForeground(UIApplication application)
{
    //remove 'background' image
    var backgroundView = this.window.ViewWithTag(1234);
    if(null != backgroundView)
        backgroundView.RemoveFromSuperview();
}
```

Steuern des Screenshots im iOS Multitasking Switcher online lesen: https://riptutorial.com/de/xamarin-ios/topic/8681/steuern-des-screenshots-im-ios-multitasking-switcher

# Kapitel 18: Suchleiste zu UITableView hinzufügen

## Bemerkungen

Objektreferenzen:

UITableView-Tabelle;
TableSource tableSource;
Listet tableItems auf;
UISearchBar searchBar;

Um das vollständige Beispiel zu untergliedern : https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/SearchBarWithTableView

## Examples

### Fügen Sie UISearchBar zu UITableView hinzu

```
public override void ViewDidLoad()
{
    base.ViewDidLoad();
    // Perform any additional setup after loading the view, typically from a nib.

    //Declare the search bar and add it to the header of the table
    searchBar = new UISearchBar();
    searchBar.SizeToFit();
    searchBar.AutocorrectionType = UITextAutocorrectionType.No;
    searchBar.AutocapitalizationType = UITextAutocapitalizationType.None;
    searchBar.TextChanged += (sender, e) =>
    {
        //this is the method that is called when the user searches
        searchTable();
    };

    Title = "SearchBarWithTableView Sample";
    table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
    //table.AutoresizingMask = UIViewAutoresizing.All;
    tableItems = new List<TableItem>();

    tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
    tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
    tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
    tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
    tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
    tableSource = new TableSource(tableItems);
    table.Source = tableSource;
    table.TableHeaderView = searchBar;
    Add(table);
}
```

```
private void searchTable()
{
    //perform the search, and refresh the table with the results
    tableSource.PerformSearch(searchBar.Text);
    table.ReloadData();
}
```

Die TableSource-Klasse sieht folgendermaßen aus:

```
public class TableSource : UITableViewSource
{
    private List<TableItem> tableItems = new List<TableItem>();
    private List<TableItem> searchItems = new List<TableItem>();
    protected string cellIdentifier = "TableCell";

    public TableSource(List<TableItem> items)
    {
        this.tableItems = items;
        this.searchItems = items;
    }

    public override nint RowsInSection(UITableView tableview, nint section)
    {
        return searchItems.Count;
    }

    public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
    {
        // request a recycled cell to save memory
        UITableViewCell cell = tableView.DequeueReusableCell(cellIdentifier);


        var cellStyle = UITableViewCellStyle.Default;

        // if there are no cells to reuse, create a new one
        if (cell == null)
        {
            cell = new UITableViewCell(cellStyle, cellIdentifier);
        }

        cell.TextLabel.Text = searchItems[indexPath.Row].Title;
        cell.ImageView.Image = UIImage.FromFile("Images/" +
searchItems[indexPath.Row].ImageName);

        return cell;
    }

    public override nint NumberOfSections(UITableView tableView)
    {
        return 1;
    }

    public void PerformSearch(string searchText)
    {
        searchText = searchText.ToLower();
        this.searchItems = tableItems.Where(x =>
x.Title.ToLower().Contains(searchText)).ToList();
    }
}
```

Suchleiste zu UITableView hinzufügen online lesen: https://riptutorial.com/de/xamarin-ios/topic/6540/suchleiste-zu-uitableview-hinzufugen

# Kapitel 19: UIImageView-Zoom in Kombination mit UIScrollView

## Bemerkungen

Die UIImageView muss sich innerhalb einer Scrollansicht befinden, damit dies funktioniert.

Die DoubleTap-Methode wechselt zwischen minScale und doubleTapScale.

## Examples

### Doppeltippen Sie auf

```
private float minScale = 1f;
private float doubleTapScale = 2f;
private float maxScale = 4f;

private void SetUpDoubleTapZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;
    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    var doubleTap = new UITapGestureRecognizer(OnDoubleTap)
    {
        NumberOfTapsRequired = 2
    };

    scrollView.AddGestureRecognizer(doubleTap);
}

private void OnDoubleTap(UIGestureRecognizer gesture)
{
    scrollView.ZoomScale = (scrollView.ZoomScale.Equals(minScale)) ? doubleTapScale :
minScale;
}
```

### Prise Geste Zoom

```
private float minScale = 1f;
private float maxScale = 4f;

private void SetUpPinchGestureZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;

    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    scrollView.ViewForZoomingInScrollView += (UIScrollView sv) => { return imageViewToZoom; };
}
```

UIImageView-Zoom in Kombination mit UIScrollView online lesen:
https://riptutorial.com/de/xamarin-ios/topic/6346/uiimageview-zoom-in-kombination-mit-uiscrollview

# Kapitel 20: Verbindung mit Microsoft Cognitive Services

## Bemerkungen

In diesem Beispiel haben wir das NuGet-Paket Microsoft.ProjectOxford.Vision verwendet: https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

Weitere Informationen zu Microsoft Cognitive Services finden Sie in der offiziellen Dokumentation: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

Das hochgeladene Sample finden Sie auf meinem GitHub: https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/XamariniOS_CognitiveServices

Ich füge auch einen Link zu meinem Blog hinzu, auf dem ich die Verwendung von Cognitive Services mit der Anwendung Xamarin Forms vorgestellt habe: http://mobileprogrammer.pl

## Examples

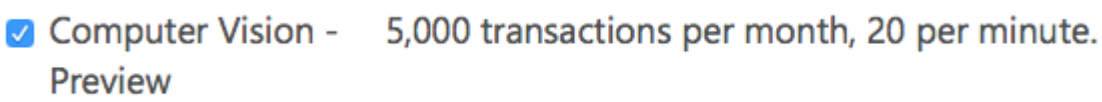
### Verbindung mit Microsoft Cognitive Services

In diesem Beispiel erfahren Sie, wie Sie Microsoft Cognitive Services mit der mobilen Xamarin iOS-Anwendung verwenden. Wir werden die Computer Vision-API verwenden, um zu erkennen, was auf dem Bild zu sehen ist.

Nachdem Sie ein Xamarin.iOS-Projekt erstellt haben, fügen Sie das NuGet-Paket zum Projekt hinzu:

https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

Mit dieser Bibliothek können wir Cognitive Services in unserer iOS-App nutzen. Ich gehe davon aus, dass Sie bereits ein Microsoft-Konto für die Verwendung registriert haben und Computer Vision Api wie auf dem Bildschirm unten aktiviert haben:



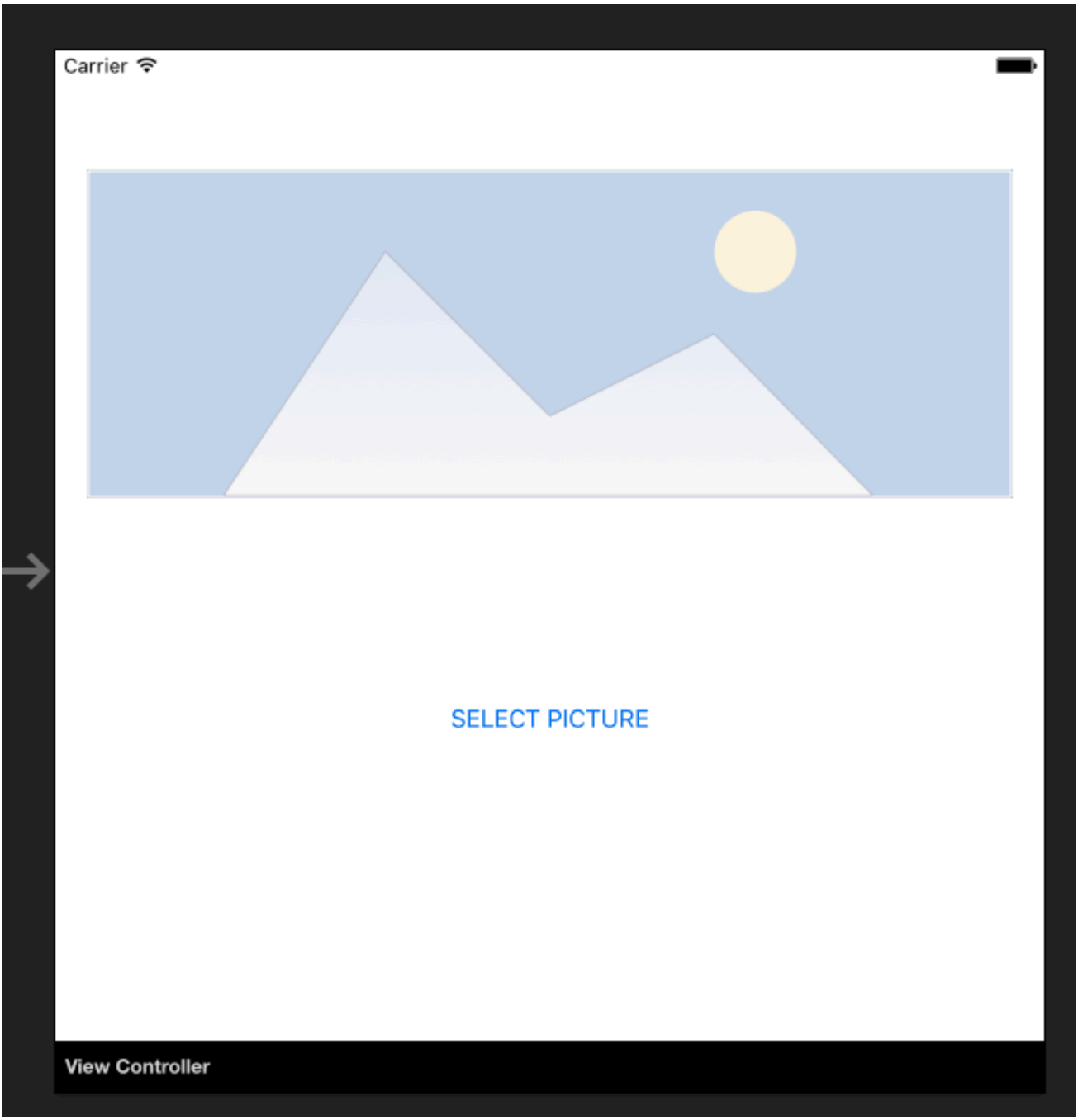

Wenn Sie unten auf "Abonnieren" klicken, wird der Api-Schlüssel generiert:

| | | |
|---|---|---|
| Computer Vision - Preview | 5,000 transactions per month, 20 per minute. | Key 1:<br>XXXXXXXXXXXXXXXXXXXXXXXXXXXX<br>Regenerate \| Show \| Copy<br>Key 2:<br>XXXXXXXXXXXXXXXXXXXXXXXXXXXX<br>Regenerate \| Show \| Copy |

Jetzt können wir den Zugriff auf Cognitive Services über die iOS-App konfigurieren. Zunächst müssen wir uns ein Bild für die Analyse machen. Dazu können wir die Xamarin Media Component verwenden, die unter folgender Adresse verfügbar ist:
https://components.xamarin.com/view/mediaplugin

Nach der erfolgreichen Installation erstellen wir eine einfache Benutzeroberfläche mit dem Bild und der Schaltfläche, um ein Bild aus der Galerie auszuwählen. Die Größe der Steuerelemente liegt bei Ihnen.

Öffnen Sie Main.storyboard und fügen Sie UIImageView hinzu, und UIButton-Steuerelemente führen den standardmäßigen ViewController aus. Fügen Sie ihnen Namen hinzu: "SelectedPictureImageView" und "SelectButton":




Jetzt sollten wir den Event-Handler "Touch Up Inside" hinzufügen, um die Bildauswahl durchzuführen:

```
partial void SelectButtonClick(UIButton sender)
{
    selectImage();
}

async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
}
```

Jetzt möchten wir die Analyseinformationen anzeigen, sobald Cognitive Services die

Informationen zurückgibt. Fügen Sie unter dem Button "AnalysisLabel" ein Label hinzu:

SELECT PICTURE

Analysis result....

Es ist Zeit, die Computer Vision-API anzuschließen!

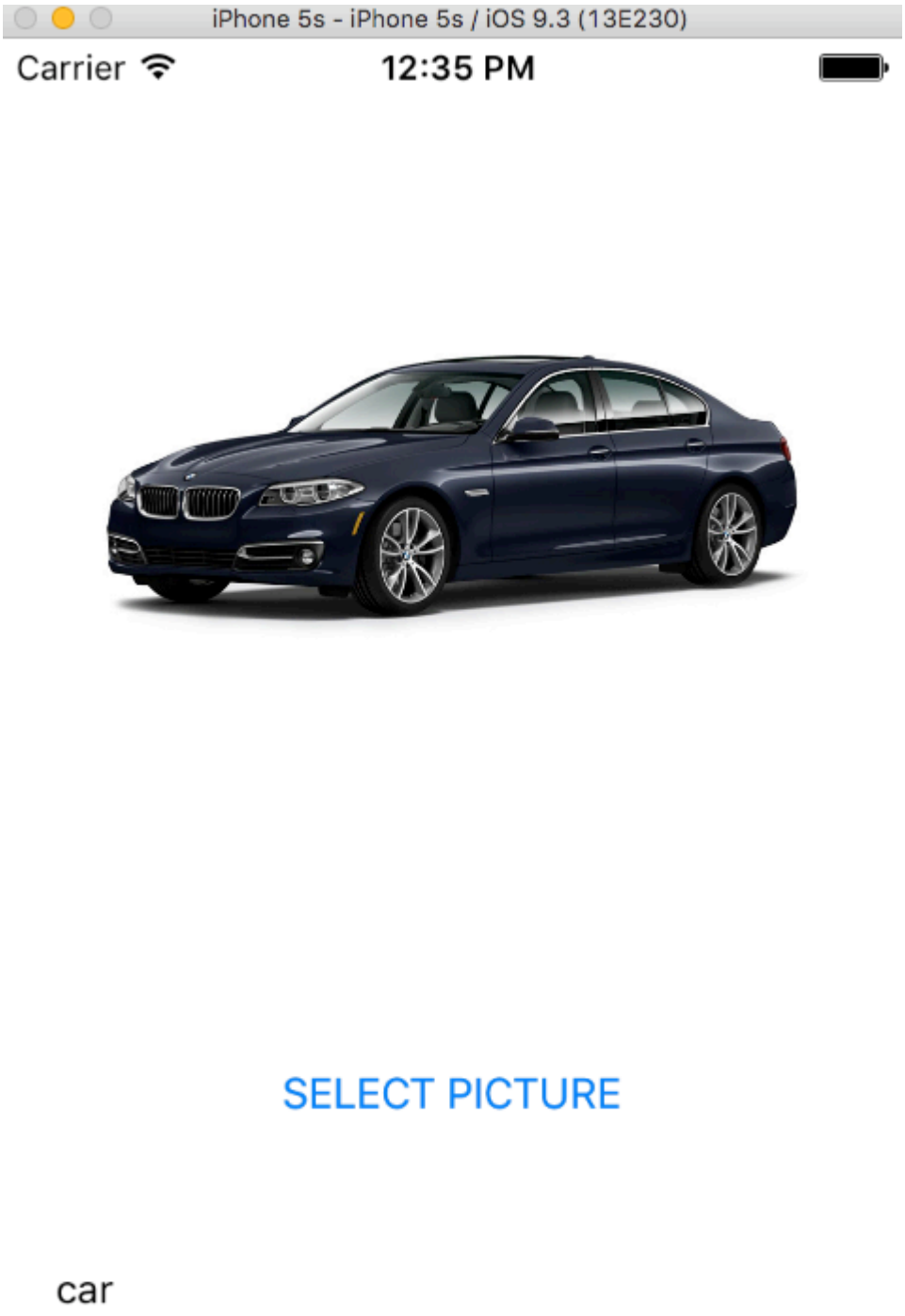
Um Informationen über das ausgewählte Bild zu erhalten, fügen Sie die Methode unten hinzu. Denken Sie daran, Ihren API-Schlüssel einzufügen!

```
async Task analyseImage(Stream imageStream)
{
    try
    {
        VisionServiceClient visionClient = new VisionServiceClient("<<YOUR API KEY HERE>>");
        VisualFeature[] features = { VisualFeature.Tags, VisualFeature.Categories,
VisualFeature.Description };
        var analysisResult = await visionClient.AnalyzeImageAsync(imageStream,
features.ToList(), null);
        AnalysisLabel.Text = string.Empty;
        analysisResult.Description.Tags.ToList().ForEach(tag => AnalysisLabel.Text =
AnalysisLabel.Text + tag + "\n");
    }
    catch (Microsoft.ProjectOxford.Vision.ClientException ex)
    {
        AnalysisLabel.Text = ex.Error.Message;
    }
}
```

Jetzt können Sie es in der "selectImage" -Methode aufrufen:

```
async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
    await analyseImage(selectedImage.GetStream());
}
```

Sobald Sie ein Bild ausgewählt haben, wird es von Microsoft Cognitive Services analysiert und das Ergebnis zurückgegeben:




Denken Sie daran, dass Ihr Bild nicht zu groß sein kann. In diesem Fall erhalten Sie folgende Informationen:

SELECT PICTURE

Input image is too large.

Es gibt viele andere Dienste, die Sie verwenden können. Weitere Informationen finden Sie in der offiziellen Dokumentation (Link im Anhang).

Verbindung mit Microsoft Cognitive Services online lesen: https://riptutorial.com/de/xamarin-ios/topic/6122/verbindung-mit-microsoft-cognitive-services

# Kapitel 21: Verwenden von iOS-Asset-Katalogen zum Verwalten von Bildern

## Bemerkungen

Mit Asset-Katalogen können Sie mehrere Auflösungen von iOS-Image-Assets verwalten. Um optimale Bilder anzuzeigen, verwendet iOS je nach Bildschirmdichte des Geräts 1x, 2x und 3x Versionen. Die 1x-Version ist nur für sehr alte, nicht auf der Netzhaut befindliche Geräte, also nicht für Apps, die nur iOS 9 unterstützen.

Asset-Kataloge unterstützen App-Ausdünnung und Slicing und optimieren die Ressourcen, die Benutzer herunterladen müssen, um eine App aus dem App Store zu installieren.

## Examples

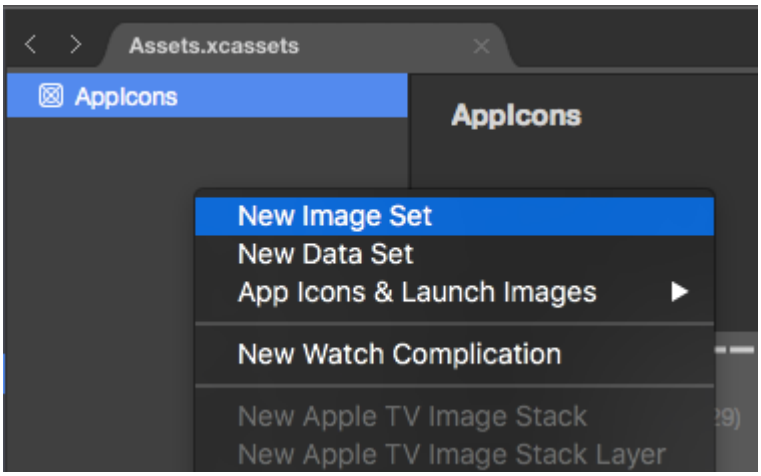
### Laden eines Bestandskatalogbildes

Ein Bild aus einem Asset-Katalog mit `UIImage.FromBundle(string imageName)`

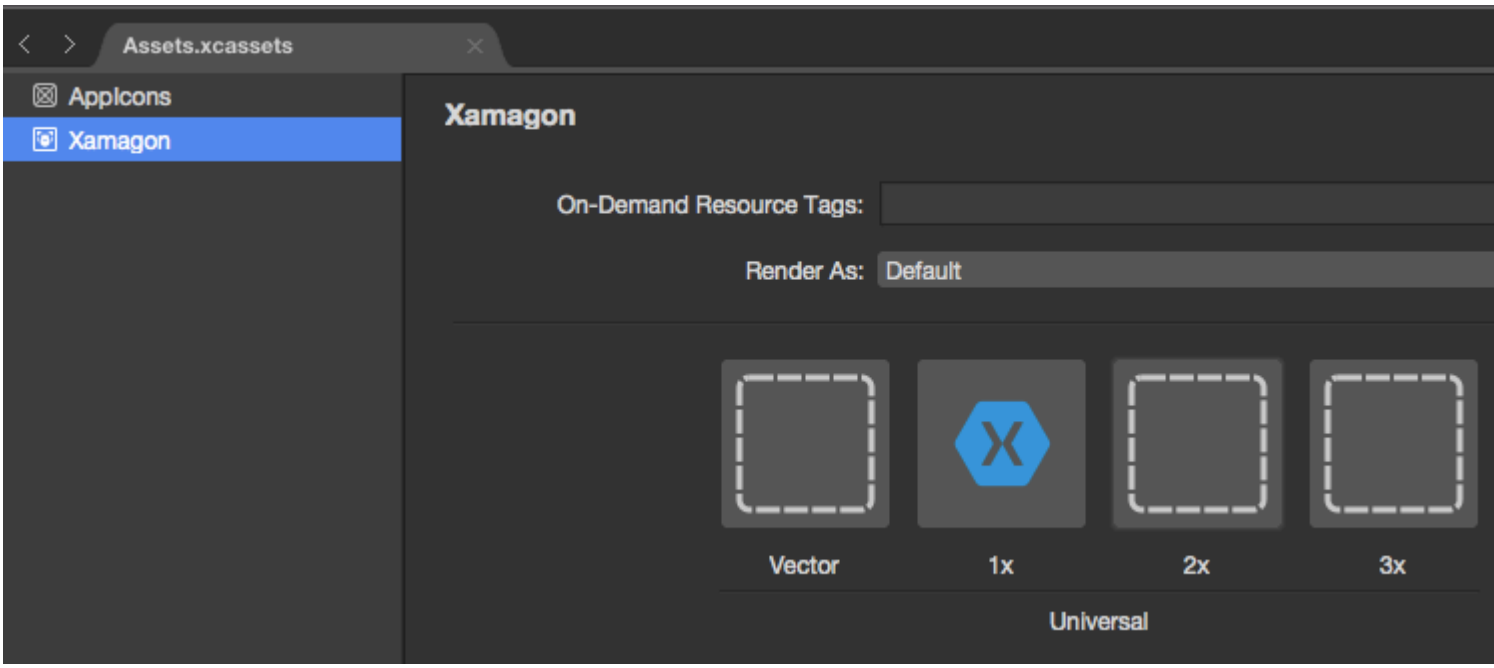
```
UIImage image = UIImage.FromBundle("ImageName");
// use the name of the image set from the asset catalog
```

Sie können das Bild für eine `UIImageView` oder für andere `UIImageView` verwenden.

### Bilder in einem Asset-Katalog verwalten




Asset-Kataloge ermöglichen die Verwaltung von Bildern, App-Symbolen und das Starten von Bildern. Image Set wird für Bilder verwendet, die in der App angezeigt werden. Universalbilder sind normalerweise die beste Option. Sie können entweder ein vektorbasiertes Bild (z. B. PDF) verwenden, das für alle Bildschirme skaliert werden kann, oder eine 1x-, 2x- und 3x-Variante verwenden. IOS wählt die entsprechende Version des Bildes für das aktuelle Gerät des Benutzers aus.




Sie können den Namen eines Sets im Asset-Katalog ändern, indem Sie auf den Namen doppelklicken. Bilder können entweder durch Ziehen und Ablegen hinzugefügt werden oder auf das Bild klicken, das Sie für eine Dateiauswahl ausfüllen möchten.

## Hinzufügen von Asset Catalog-Bildern im Storyboard

Bestandskatalogbilder können von Storyboards wie jede andere Art von Bildern verwendet werden, die dem Projekt hinzugefügt werden. Sie werden automatisch als Option in `UIImageView` und anderen Ansichten `UIImageView` die das Hinzufügen eines Bildes unterstützen.

Verwenden von iOS-Asset-Katalogen zum Verwalten von Bildern online lesen: https://riptutorial.com/de/xamarin-ios/topic/6241/verwenden-von-ios-asset-katalogen-zum-verwalten-von-bildern

# Kapitel 22: Xamarin iOS Google Places Autovervollständigung

## Einführung

Seit ich angefangen habe, mit Xamarin zu arbeiten, gab es viele Dinge, von denen ich wünschte, jemand anderes hätte dies bereits dokumentiert. Hier erkläre ich, wie man einen Aspekt der automatischen Vervollständigung von Google Places verwendet, wobei das UI-Steuerelement einen Ergebniscontroller verwendet. Obwohl dieser Code auf Googles eigenen Beispielen basiert, die von Swift in C # konvertiert wurden, habe ich mein Bestes gegeben, um sicherzustellen, dass es ein voll funktionsfähiges Beispiel ist. Ich hoffe sehr, dass diese Dokumentation anderen helfen wird.

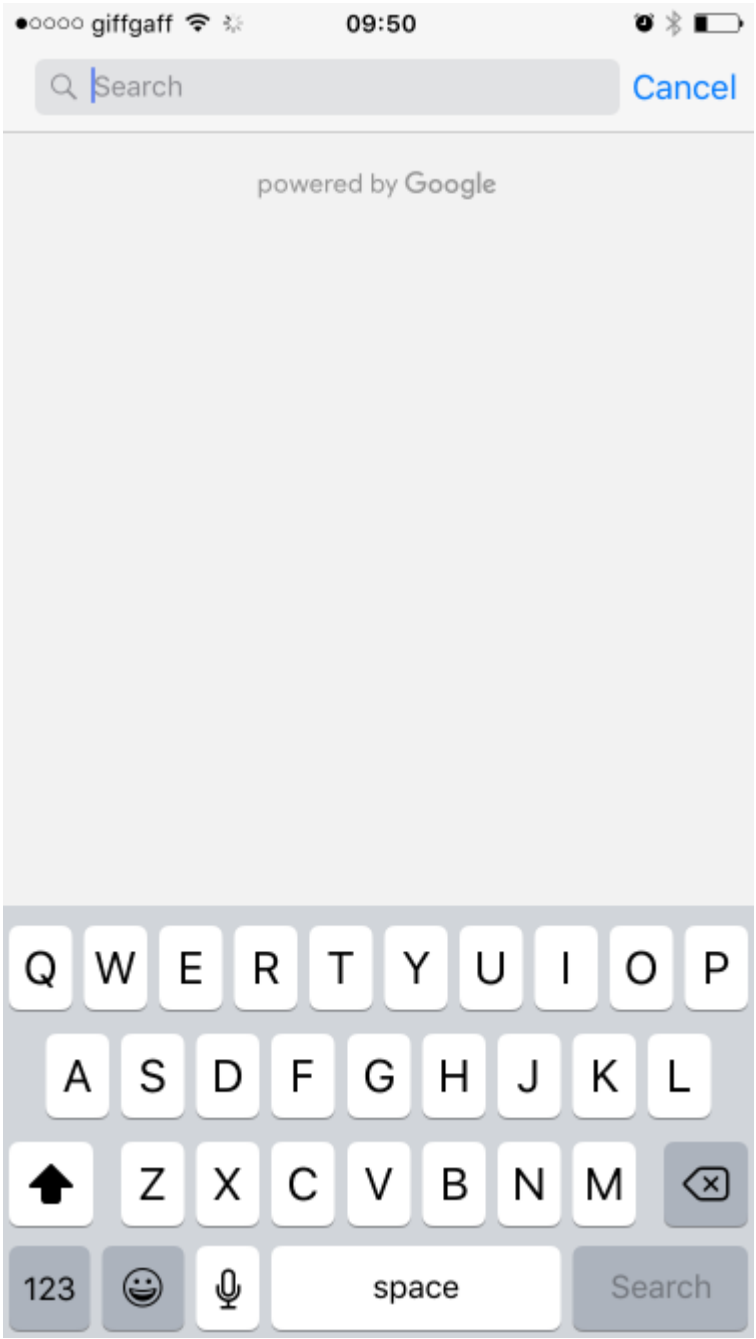
## Examples

### Fügen Sie eine Autocomplete-UI-Steuerung mit Ergebnissteuerung hinzu.

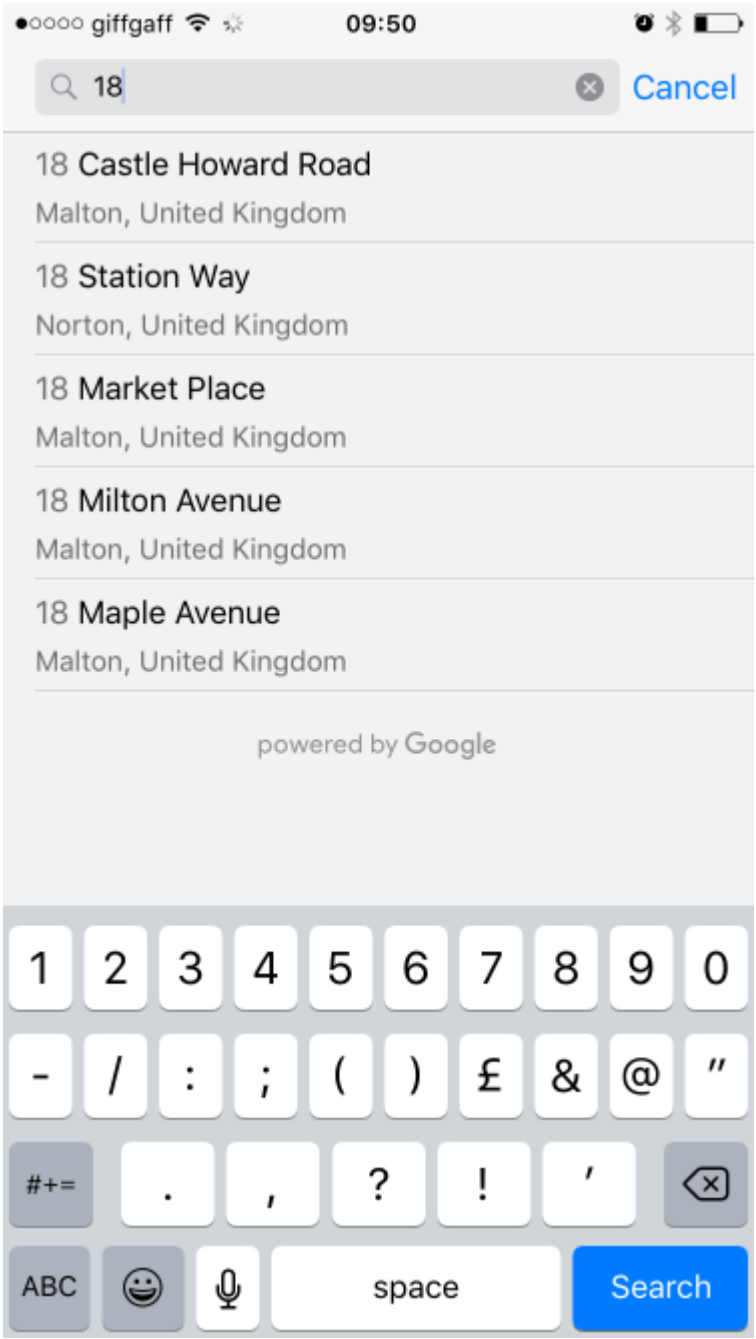
Das Autocomplete-UI-Steuerelement ist ein Suchdialog mit integrierter Autocomplete-Funktion. Wenn ein Benutzer Suchbegriffe eingibt, zeigt das Steuerelement eine Liste der vorhergesagten Orte zur Auswahl an. Wenn der Benutzer eine Auswahl trifft, wird eine GMSPlace-Instanz (In Xamarin platzieren) zurückgegeben, die Ihre App verwenden kann, um Details zum ausgewählten Ort abzurufen.

Wie oben erwähnt, verwendet dieses Beispiel einen Ergebniscontroller, der eine bessere Kontrolle über die Texteingabe-Benutzeroberfläche ermöglicht. Der Ergebniscontroller schaltet die Sichtbarkeit der Ergebnisliste basierend auf dem Fokus der Eingabe-Benutzeroberfläche dynamisch um.

Ziel dieses Codes ist es, einen Bildschirm wie folgt anzuzeigen:




Dies wird Ihre Adresse automatisch vervollständigen, wenn Sie mit der Eingabe beginnen, wie im Bild unten:




**Anleitung:**

1. Zuerst müssen Sie die Google Maps-API zu unserem Visual Studio hinzufügen. Sie ist über Nuget verfügbar. Suchen Sie einfach nach Xamarin.Google.iOS.Maps, fügen Sie sie Ihrem iOS-Projekt hinzu. Alternativ können Sie sie auch von Xamarin herunterladen. Google Maps iOS SDK für Google Maps

2. Wir brauchen so etwas wie eine Schaltfläche, um den Google Autocomplete View Controller auszulösen. In diesem Beispiel mache ich dies mit einem Storyboard und habe eine Schaltfläche mit dem Namen GoogleButton hinzugefügt. Sie könnten sie mit Code auslösen, es spielt keine Rolle.

3. Fügen Sie unter ViewDidLoad in Ihrer View-Controller-Klasse den folgenden Code hinzu. In meinem Beispiel verwende ich nicht den tatsächlichen Standort der mobilen Geräte, meine endgültige Lösung würde dies natürlich tun, aber dies war ein Test, und ich wollte keinen

zusätzlichen Code implementieren, bis ich bewiesen habe, dass dies funktioniert oder verwässert wurde, was ich versuche um dir zu zeigen:

// Code, um den Controller für die automatische vollständige Ansicht von Google Places aufzurufen.

```
GoogleButton.TouchUpInside += (sender, ea) =>
{

    var FakeCoordinates = new CLLocationCoordinate2D()
    {
        Latitude = 54.135364,
        Longitude = -0.797888
    };

    var north = LocationWithBearing(45, 3000, FakeCoordinates);
    var east = LocationWithBearing(225, 3000, FakeCoordinates);

    var autocompleteController = new AutocompleteViewController();
    autocompleteController.Delegate = new AutoCompleteDelegate();
    autocompleteController.AutocompleteBounds = new CoordinateBounds(north, east);
    PresentViewController(autocompleteController, true, null);

};
```

4. Dies ist optional, aber ich habe eine Funktion hinzugefügt, um die lokalen Grenzen zu berechnen. Ich überschreite 3000. Diese Zahl wird in Metern angegeben. Wenn Sie also größere Anfangsgrenzen wünschen, passen Sie diese bitte an. Beachten Sie bitte, dass bei der Google-Suche nach Adressen gesucht wird in der Welt gewichtet es die ersten Ergebnisse nur an diese Grenzen. Diese Funktion wurde aus einem Stack-Überlaufpfosten entlehnt. Ich habe sie für unsere Zwecke von Swift nach C # konvertiert:

```
public CLLocationCoordinate2D LocationWithBearing(Double bearing, Double distanceMeters,
CLLocationCoordinate2D origin)
{
    var distRadians = distanceMeters/(6372797.6);

    var rbearing = bearing*Math.PI/180.0;

    var lat1 = origin.Latitude*Math.PI/180;
    var lon1 = origin.Longitude*Math.PI/180;

    var lat2 = Math.Asin(Math.Sin(lat1)*Math.Cos(distRadians) +
Math.Cos(lat1)*Math.Sin(distRadians)*Math.Cos(rbearing));
    var lon2 = lon1 + Math.Atan2(Math.Sin(rbearing)*Math.Sin(distRadians)*Math.Cos(lat1),
                    Math.Cos(distRadians) - Math.Sin(lat1)*Math.Sin(lat2));

    return new CLLocationCoordinate2D(latitude: lat2*180/ Math.PI, longitude:
lon2*180/Math.PI);
}
```

5. Dieses letzte Code-Snippet ist der Delegierte für die automatische Vervollständigung. Wir benötigen diesen Delegierten, um alle Google-Aktionen abzuwickeln:

```
public  class AutoCompleteDelegate : AutocompleteViewControllerDelegate
```

```
{

    public override void DidFailAutocomplete(AutocompleteViewController viewController,
NSError error)
    {
        // TODO: handle the error.
        Debug.Print("Error: " + error.Description);
    }


    public override void DidAutocomplete(AutocompleteViewController viewController, Place
place)
    {
        Debug.Print(place.Name);
        Debug.Print(place.FormattedAddress);

        viewController.DismissViewController(true, null);
    }

    public override void DidRequestAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void DidUpdateAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void WasCancelled(AutocompleteViewController viewController)
    {
        viewController.DismissViewController(true, null);
    }

}
```

Führen Sie Ihr Projekt aus und es sollte einwandfrei funktionieren. Dieses Beispiel ist ziemlich fokussiert, aber hoffentlich wird Ihnen ein grundlegendes Beispiel dafür gegeben, wie alle Steuerelemente der Google Autocomplete-Benutzeroberfläche funktionieren müssen. Vielen Dank!

Xamarin iOS Google Places Autovervollständigung online lesen: https://riptutorial.com/de/xamarin-ios/topic/9041/xamarin-ios-google-places-autovervollstandigung

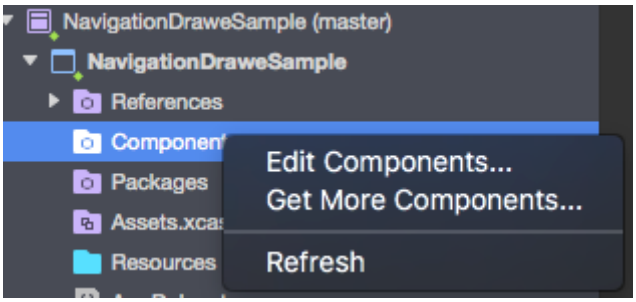
# Kapitel 23: Xamarin.iOS Navigationsschublade

## Syntax

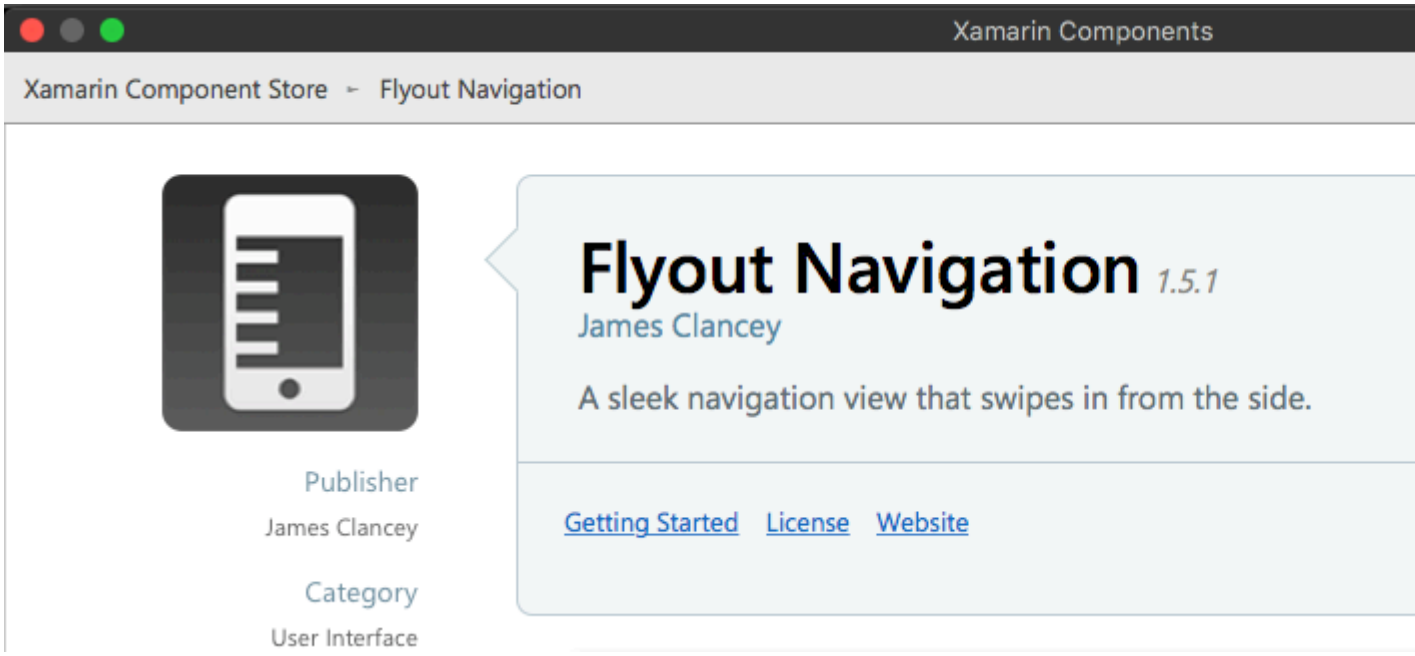
1. Flayout-Navigationskomponente: https://components.xamarin.com/view/flyoutnavigation

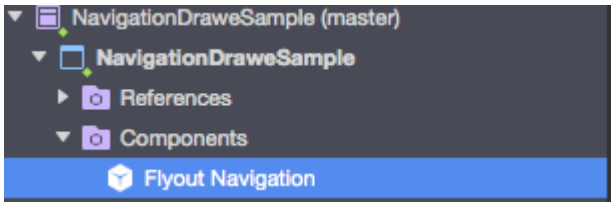
## Examples

### Xamarin.iOS Navigationsschublade

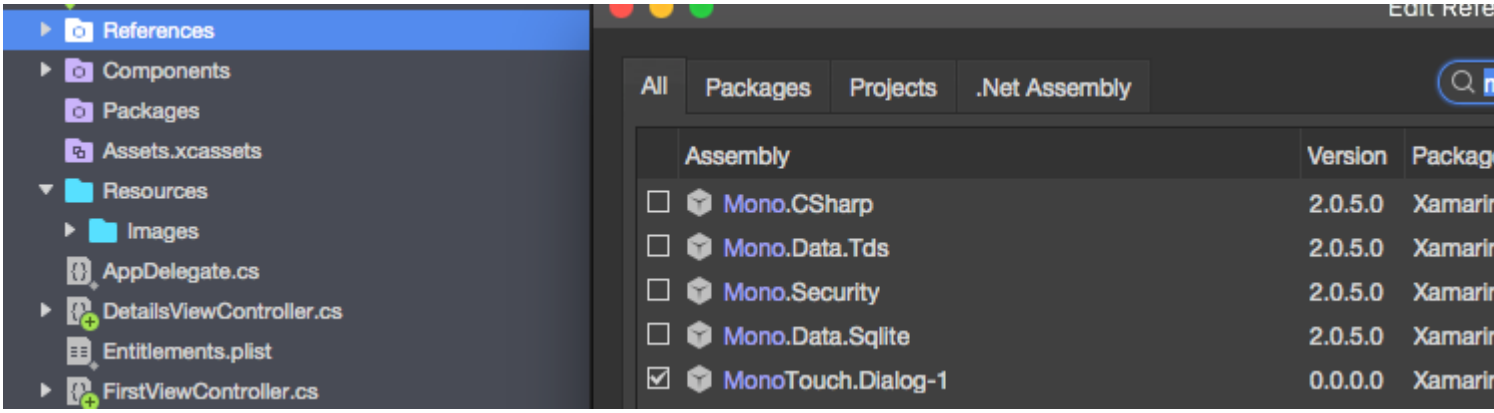
1. Erstellen Sie ein neues leeres Xamarin.iOS-Projekt (Single View App).
2. Klicken Sie mit der rechten Maustaste auf den Ordner "Components" und wählen Sie "Get More Components" aus. 

3. Geben Sie im Suchfeld "Flout Navigation" ein und fügen Sie Ihrer App die folgende Komponente hinzu:

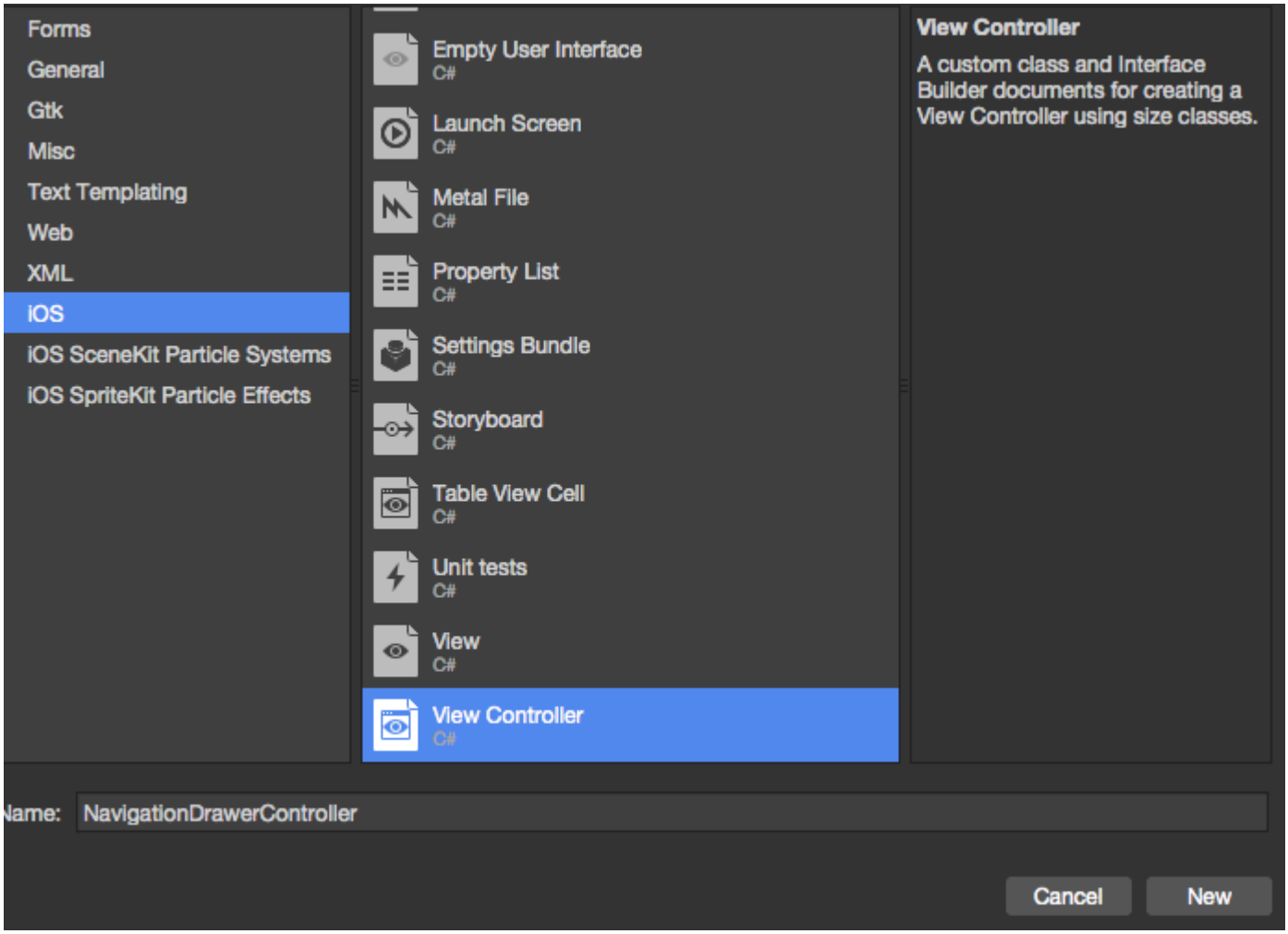





Denken Sie auch daran, die Referenz "Mono.Touch.Dialog-1" hinzuzufügen:




4. Klicken Sie nun das Projekt an und fügen Sie einen neuen UIViewController namens "NavigationDrawerController" hinzu:




5. Der Code für die "NavigationDrawerController" -Klasse sollte jetzt wie folgt aussehen:

```
public partial class NavigationDrawerController : UIViewController
```

```
{
    public NavigationDrawerController(IntPtr handle) : base(handle)
    {
    }

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

        NavigationItem.LeftBarButtonItem = getMenuItem();
        NavigationItem.RightBarButtonItem = new UIBarButtonItem { Width = 40 };
    }

    UIBarButtonItem getMenuItem()
    {
        var item = new UIBarButtonItem();
        item.Width = 40;
        //Please provide your own icon or take mine from the GitHub sample:
        item.Image = UIImage.FromFile("Images/menu_button@2x.png");
        item.Clicked += (sender, e) =>
        {
            if (ParentViewController is MainNavigationController)
                (ParentViewController as MainNavigationController).ToggleMenu();
        };

        return item;
    }
}
```

Keine Sorge, dass "MainNavigationController" rot markiert ist - wir werden es im nächsten Schritt hinzufügen.

6. Öffnen Sie nun die Datei "Main.storyboard":

a) Fügen Sie einen UIViewController hinzu:

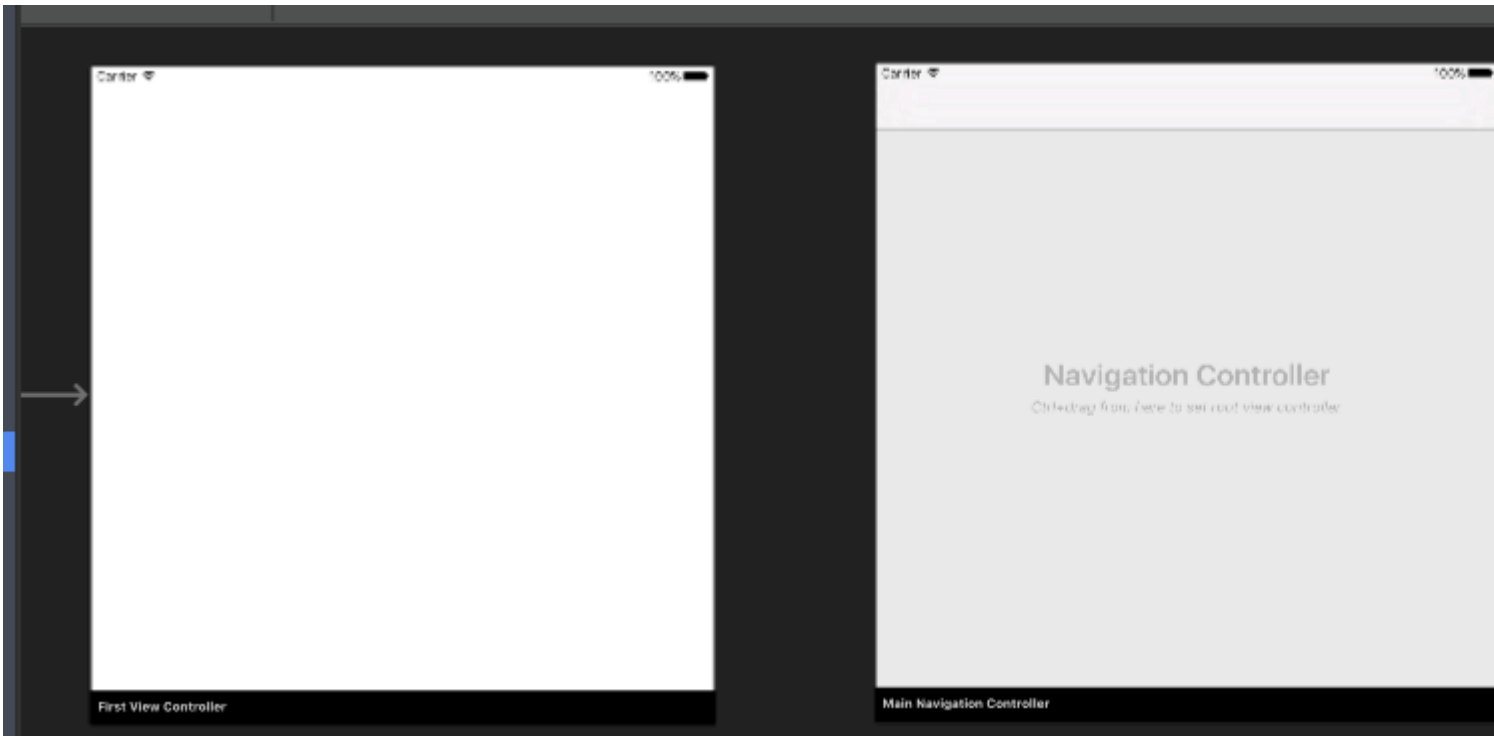
Füllen Sie die Felder "Klasse" und "StoryboardID" mit diesem Namen: "FirstViewController"

b) Fügen Sie anschließend den Navigationscontroller mit dem Root-UIViewController hinzu:

Füllen Sie die Felder "Class" und "StoryboardID" mit diesem Namen: "MainNavigationController" für den Navigationscontroller

Füllen Sie die Felder "Class" und "StoryboardID" mit diesem Namen: "DetailsViewController" für den Root-Controller

Xamarin (oder Visual) Studio erstellt Code-Behind-Klassen für die oben genannten Controller.




7. Öffnen Sie nun die Klasse "FirstViewController" und fügen Sie den folgenden Code ein:

```
public partial class FirstViewController : UIViewController
{
    public FirstViewController (IntPtr handle) : base (handle)
    {
    }

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

    void createNavigationFlyout()
    {
        var navigation = new FlyoutNavigationController
        {
            //Here are sections definied for the drawer:
            NavigationRoot = new RootElement("Navigation")
            {
                new Section ("Pages")
                {
                    new StringElement ("MainPage")
                }
            },

            //Here are controllers definied for the drawer (in this case navigation controller
with one root):
             ViewControllers = new[]
             {

(MainNavigationController)Storyboard.InstantiateViewController("MainNavigationController")
             }
        };

        View.AddSubview(navigation.View);
```

```
    }
}
```

8. Öffnen Sie die "MainNavigationController" -Klasse und fügen Sie den folgenden Code ein:

```
public partial class MainNavigationController : UINavigationController
{
     public MainNavigationController (IntPtr handle) : base (handle)
     {
     }
     //Responsible for opening/closing drawer:
     public void ToggleMenu()
     {
         if (ParentViewController is FlyoutNavigationController)
         (ParentViewController as FlyoutNavigationController).ToggleMenu();
     }
 }
```

9. Letzte Klasse namens "DetailsViewController" sollte folgendermaßen aussehen:

```
public partial class DetailsViewController : NavigationDrawerController
{
    public DetailsViewController (IntPtr handle) : base(handle)
    {
    }
}
```

Bitte beachten Sie, dass "DetailsViewController" von "NavigationDrawerController" stammt, das wir zu Beginn erstellt haben.

Das ist es. Jetzt können Sie die Schublade beliebig anpassen. Bitte finden Sie auch ein fertiges Sample auf meinem GitHub:

https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/Xamarin.iOS.NavigationDrawer

Xamarin.iOS Navigationsschublade online lesen: https://riptutorial.com/de/xamarin-ios/topic/6574/xamarin-ios-navigationsschublade

# Credits

| S. No | Kapitel | Contributors |
|---|---|---|
| 1 | Erste Schritte mit Xamarin.iOS | Amy Burns, chrisntr, Community, Dominic, hankide, Sergey, valdetero |
| 2 | Alarme | chrisntr, Gil Sand, patridge, Pilatus, Prashant C, valdetero |
| 3 | Arbeiten mit Xib und Storyboards in Xamarin.iOS | lukya |
| 4 | Asset-Kataloge verwenden | aniket.ghode, mnoronha |
| 5 | Automatisches Layout in Xamarin.iOS | ben, patridge, Tom Hawkin, valdetero |
| 6 | Berechnung der variablen Zeilenhöhe in GetHeightForRow | Larry OBrien, valdetero |
| 7 | Berührungsidentifikation | Amy Burns, ben, DannyC, Matthew, Peter Zhong, valdetero |
| 8 | Best Practices für die Migration von UILocalNotification zu User Notifications Framework | Aditya Kumar |
| 9 | Binden Sie schnelle Bibliotheken | Alex Sorokoletov, Elad Nava, Esam Sherif, J. Rahmati, James Mundy, Lucas Teixeira |
| 10 | Erstellen und Verwenden von benutzerdefinierten Prototyp-Tabellenzellen in xamarin.iOS mithilfe des Storyboards | Daniel Krzyczkowski, valdetero |
| 11 | Fügen Sie PullToRefresh zu UITableView hinzu | Aditya Kumar, valdetero |
| 12 | Gleichzeitiges | Ashan, Pilatus, Tom Gilder, valdetero |

| | | |
|---|---|---|
| | Programmieren in Xamarin.iOS | |
| 13 | Hinzufügen von UIRefreshControl zu einer Tabellensicht | manishKungwani, valdetero |
| 14 | Methoden zur Größenänderung für UIImage | Frauke Nonnenmacher, raymondis, valdetero |
| 15 | So verwenden Sie Asset-Asset-Kataloge | Aditya Kumar |
| 16 | Steuern des Screenshots im iOS Multitasking Switcher | ben |
| 17 | Suchleiste zu UITableView hinzufügen | Aditya Kumar, valdetero |
| 18 | UIImageView-Zoom in Kombination mit UIScrollView | Citroenfris, valdetero |
| 19 | Verbindung mit Microsoft Cognitive Services | Daniel Krzyczkowski, valdetero |
| 20 | Verwenden von iOS-Asset-Katalogen zum Verwalten von Bildern | dylansturg, valdetero |
| 21 | Xamarin iOS Google Places Autovervollständigung | Conrad |
| 22 | Xamarin.iOS Navigationsschublade | Daniel Krzyczkowski, valdetero |