



eBook Gratuit

APPRENEZ Xamarin.iOS

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

[#xamarin.io](https://twitter.com/xamarin)

S

Table des matières

À propos.....	1
Chapitre 1: Démarrer avec Xamarin.iOS.....	2
Remarques.....	2
Versions.....	2
Exemples.....	2
Commencez avec Xamarin Studio.....	2
Commencer dans Visual Studio.....	6
Bonjour le monde.....	12
Chapitre 2: Ajout de UIRefreshControl à une vue de table.....	16
Exemples.....	16
Ajouter un UIRefreshControl à une TableView.....	16
Chapitre 3: Ajout de UIRefreshControl à une vue de table.....	17
Exemples.....	17
Ajouter un UIRefreshControl simple à un UIScrollView.....	17
Style 1:.....	17
Style 2:.....	17
Style 3:.....	17
Chapitre 4: Ajouter PullToRefresh à UITableView.....	19
Remarques.....	19
Exemples.....	19
Ajout de UIRefreshControl à UITableView.....	19
Chapitre 5: Ajouter une barre de recherche à UITableView.....	21
Remarques.....	21
Exemples.....	21
Ajouter UISearchBar à UITableView.....	21
Chapitre 6: Calcul de la hauteur des lignes variables dans GetHeightForRow.....	24
Remarques.....	24
Exemples.....	24
Utiliser GetHeightForRow.....	24

Chapitre 7: Comment utiliser les catalogues d'actifs d'actifs	28
Exemples.....	28
Utilisation de catalogues d'actifs.....	28
Chapitre 8: Connexion avec Microsoft Cognitive Services	29
Remarques.....	29
Exemples.....	29
Connexion avec Microsoft Cognitive Services.....	29
Chapitre 9: Contrôle de la capture d'écran dans le commutateur multitâche iOS	36
Introduction.....	36
Remarques.....	36
Exemples.....	36
Afficher une image pour l'instantané.....	36
Chapitre 10: Créer et utiliser des cellules de tableau prototype personnalisées dans xamar	37
Exemples.....	37
Créer une cellule personnalisée à l'aide de Storyboard.....	37
Chapitre 11: Des alertes	41
Exemples.....	41
Afficher une alerte.....	41
Afficher une alerte de connexion.....	41
Afficher une feuille d'action.....	42
Afficher le dialogue d'alerte modale.....	42
Chapitre 12: ID tactile	44
Paramètres.....	44
Remarques.....	44
Exemples.....	45
Ajouter un identifiant tactile à votre application.....	45
Utilisation du trousseau.....	47
Chapitre 13: Méthodes conseillées pour la migration d'UILocalNotification vers le cadre de	50
Exemples.....	50
UserNotifications.....	50
Chapitre 14: Méthodes de redimensionnement pour UIImage	51

Exemples.....	51
Redimensionner l'image - avec le rapport d'aspect.....	51
Redimensionner l'image - sans rapport d'aspect.....	51
Recadrer l'image sans redimensionner.....	51
Chapitre 15: Mise en page automatique dans Xamarin.iOS.....	53
Exemples.....	53
Ajout de contraintes avec iOS 9+ Layout Anchors.....	53
Ajout de contraintes à l'aide du langage VFL.....	53
Utilisation de Cirrious.FluentLayout.....	54
Ajout de contraintes avec la maçonnerie.....	54
Chapitre 16: Programmation concurrente dans Xamarin.iOS.....	56
Exemples.....	56
Manipulation de l'interface utilisateur à partir de threads d'arrière-plan.....	56
Utiliser Async et attendre.....	56
Chapitre 17: Reliure des bibliothèques rapides.....	58
Introduction.....	58
Remarques.....	58
Exemples.....	58
Relier une bibliothèque Swift à Xamarin.iOS.....	58
1.1 Préparez les classes Swift que vous souhaitez exporter.....	58
1.2 Construire le cadre.....	59
2. Créez une bibliothèque de graisse.....	61
3. Importer la bibliothèque.....	63
4. Créez le fichier ApiDefinition basé sur le fichier LIBRARY-Swift.h dans les en-têtes.....	64
5. Modifiez tous les protocoles [Protocol] et [BaseType] pour inclure le nom de la classe.....	65
6.1 Inclure toutes les dépendances Swift à exécuter.....	66
6.2. Identifier les dépendances Swift à inclure.....	68
7. Incluez SwiftSupport pour pousser l'application vers AppStore.....	69
Remarques.....	72
Avertissement.....	73
Chapitre 18: Tiroir de navigation Xamarin.iOS.....	74

Syntaxe.....	74
Exemples.....	74
Tiroir de navigation Xamarin.iOS.....	74
Chapitre 19: Travailler avec Xib et Storyboards dans Xamarin.iOS.....	79
Exemples.....	79
Ouverture de Xib / Storyboard dans Xcode Interface Builder à la place.....	79
Chapitre 20: UIImageView zoom en combinaison avec UIScrollView.....	80
Remarques.....	80
Exemples.....	80
Tapez deux fois.....	80
Geste de pincement zoom.....	80
Chapitre 21: Utilisation de catalogues d'actifs.....	82
Exemples.....	82
Ajout d'éléments d'image au catalogue d'actifs.....	82
Chapitre 22: Utilisation de catalogues d'actifs iOS pour gérer des images.....	85
Remarques.....	85
Exemples.....	85
Chargement d'une image de catalogue d'actifs.....	85
Gestion des images dans un catalogue d'actifs.....	85
Ajout d'images de catalogue d'actifs dans le storyboard.....	86
Chapitre 23: Xamarin iOS Google Insertion automatique.....	87
Introduction.....	87
Exemples.....	87
Ajoutez un contrôle d'auto-complétion d'interface utilisateur avec le contrôleur de résultat.....	87
Crédits.....	92

À propos

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Chapitre 1: Démarrer avec Xamarin.iOS

Remarques

Xamarin.iOS vous permet de créer des applications iOS natives en utilisant les mêmes contrôles de l'interface utilisateur que dans Objective-C et Xcode, mais avec la flexibilité et l'élégance d'un langage moderne (C #), la puissance de la bibliothèque de classes de base .NET , et deux IDE de première classe - Xamarin Studio et Visual Studio - à portée de main.

Pour plus d'informations sur l'installation de Xamarin.iOS sur votre ordinateur Mac ou Windows, reportez-vous aux guides de [démarrage](#) du centre de développement Xamarin.

Versions

Version	Date de sortie
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

Des informations détaillées sur chaque version peuvent être trouvées ici:

<https://developer.xamarin.com/releases/ios/>

Exemples

Commencez avec Xamarin Studio

1. Accédez à **Fichier> Nouveau> Solution** pour afficher la nouvelle boîte de dialogue de projet.
2. Sélectionnez **Single View App** et appuyez sur **Suivant**
3. Configurez votre application en définissant le nom et l'ID de votre application, puis appuyez sur **Suivant** :

Configure your iOS app

App Name: HelloApp

Organization Identifier: com.xamarin

Bundle Identifier: com.xamarin.helloapp



Devices: iPad

iPhone

Select the minimum iOS version you support.

5. Pour exécuter votre application, sélectionnez le débogage | Configuration de l'iPhone 6s iOS 9.x et appuyez sur le bouton **Play** :



6. Cela lancera le simulateur iOS et affichera votre application vide:

2. Naviguez vers Visual C #> iOS> iPhone et sélectionnez App vue unique:

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



Blank App (iPhone)



Master-Detail App (iPhone)



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)

▷ Online

[Click here for more](#)

Name:

HelloApp

Location:

C:\Users\Amy\Documents\

Solution name:



HelloApp

3. Donnez un **nom** à votre application et appuyez sur **OK** pour créer votre projet.
4. Sélectionnez l'icône Mac Agent dans la barre d'outils, comme illustré ci-dessous:



5. Sélectionnez le Mac qui va créer votre application à partir de la liste (assurez-vous que Mac est configuré pour recevoir la connexion!), **Puis** appuyez sur **Connecter** :

Select a Mac to use it as a Xamarin Mac Agent:

-  amyb.local
10.211.55.2
-  10.1.8.95
10.1.8.95

Add Mac...

[Where's my Mac?](#)

6. Pour exécuter votre application, sélectionnez le **débogage** | Configuration **iPhone Simulator** et appuyez sur le bouton Play:

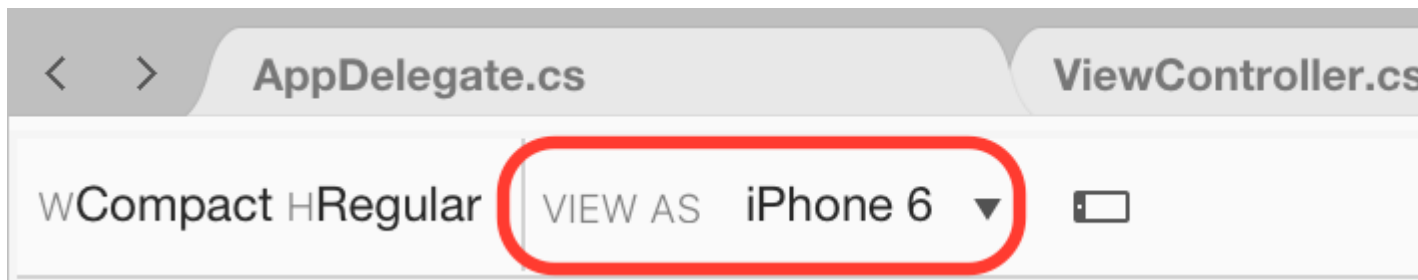
et appuyez sur le bouton Play:

et appuyez sur le bouton Play:

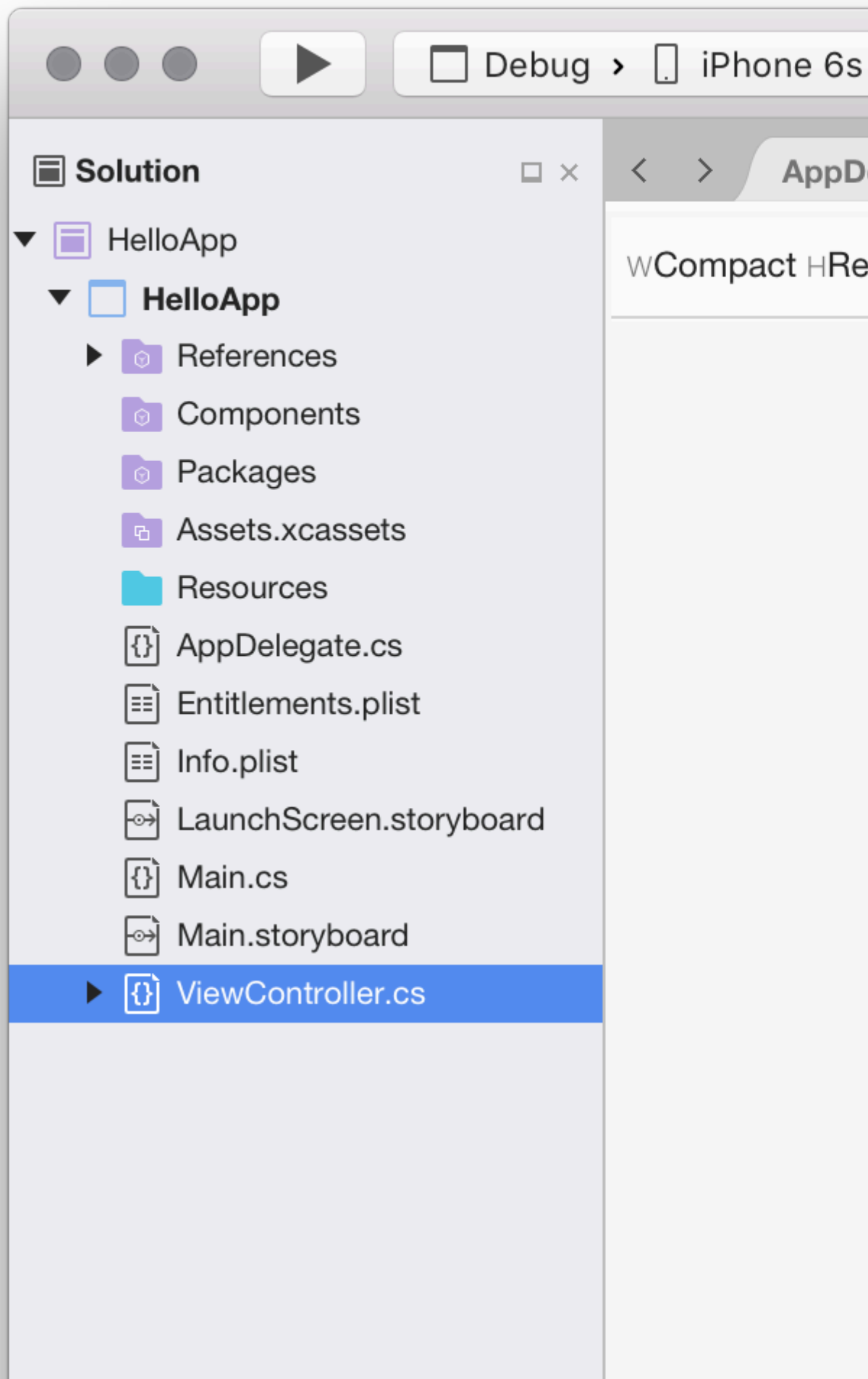


7. Cela lancera le simulateur iOS sur le Mac et affichera votre application vide:

2. Définir la **vue** de l'iPhone 6:



3. Faites glisser une étiquette et un bouton de la boîte à outils vers la surface de conception afin qu'elle ressemble à l'image ci-dessous:



4. Dans le pavé Propriétés, attribuez à l'étiquette et au bouton les propriétés suivantes:

rien	prénom	Titre
Étiquette	lblClicks	[blanc]
Bouton	clickMe	Cliquez sur moi!

5. Ajoutez le code suivant à la méthode **ViewDidLoad** dans la classe **ViewController** :

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

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

6. Exécuter l'application

Lire Démarrer avec Xamarin.iOS en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/402/demarrer-avec-xamarin-ios>

Chapitre 2: Ajout de UIRefreshControl à une vue de table

Exemples

Ajouter un UIRefreshControl à une TableView

Hypothèses:

TableView - référence à la TableView

DataSource - est une classe qui hérite de UITableViewSource

DataSource.Objects - est une liste publique <object> (), accessible à UIViewController

```
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;
}
```

Lire Ajout de UIRefreshControl à une vue de table en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/4642/ajout-de-uirefreshcontrol-a-une-vue-de-table>

Chapitre 3: Ajout de UIRefreshControl à une vue de table

Exemples

Ajouter un UIRefreshControl simple à un UIScrollView

Nous supposons une `UIScrollView` pleinement fonctionnelle nommée `_scrollView` ;

Notez que `UITableView` , `UICollectionView` sont également des scrollviews, les exemples suivants fonctionneraient donc sur ces éléments d'interface utilisateur.

Tout d'abord, création et affectation

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

Deuxièmement, connectez l'événement d'actualisation à une méthode. Il y a différentes façons de le faire.

Style 1:

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

Style 2:

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

Style 3:

```
refreshControl.ValueChanged += HandleRefreshValueChanged;

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

Troisième et dernier ajout du contrôle de rafraîchissement à notre scrollview.

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

Lire Ajout de UIRefreshControl à une vue de table en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/8371/ajout-de-uirefreshcontrol-a-une-vue-de-table>

Chapitre 4: Ajouter PullToRefresh à UITableView

Remarques

Références d'objet:

```
Table UITableView;  
TableSource tableSource;  
bool useRefreshControl = false;  
UIRefreshControl RefreshControl;  
List tableItems;
```

TableSource et TableItem sont des classes définies par l'utilisateur

Pour l'exemple complet, vous pouvez utiliser: <https://github.com/adiaditya/Xamarin.iOS-Samples/tree/master/PullToRefresh>

Exemples

Ajout de UIRefreshControl à 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
```

Lire Ajouter PullToRefresh à UITableView en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6565/ajouter-pulltorefresh-a-uitableview>

Chapitre 5: Ajouter une barre de recherche à UITableView

Remarques

Références d'objet:

```
Table UITableView;  
TableSource tableSource;  
List tableItems;  
UISearchBar searchBar;
```

Pour générer l'échantillon complet: <https://github.com/adiiditya/Xamarin.iOS-Samples/tree/master/SearchBarWithTableView>

Exemples

Ajouter UISearchBar à UITableView

```
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();
}

```

La classe TableSource ressemblera à ceci:

```

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();
    }
}

```

Lire Ajouter une barre de recherche à UITableView en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6540/ajouter-une-barre-de-recherche-a-uitableview>

Chapitre 6: Calcul de la hauteur des lignes variables dans GetHeightForRow

Remarques

Le calcul de la hauteur des lignes peut être coûteux et les performances de défilement peuvent être affectées si vous disposez de plus grandes quantités de données. Dans ce scénario, **substituez** `UITableViewSource.EstimatedHeight (UITableView, NSIndexPath)` pour fournir rapidement un nombre suffisant pour un défilement rapide, par exemple:

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

Exemples

Utiliser GetHeightForRow

Pour définir une hauteur de ligne personnalisée, remplacez

`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 ...
}
```

La classe de domaine de la table (dans ce cas, elle a 1 de 3 couleurs aléatoires et 1 de 3 hauteurs aléatoires):

```
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);
    }
}
```

Qui ressemble à:

<https://riptutorial.com/fr/xamarin-ios/topic/6515/calcul-de-la-hauteur-des-lignes-variables-dans-getheightforrow>

Chapitre 7: Comment utiliser les catalogues d'actifs d'actifs

Exemples

Utilisation de catalogues d'actifs

Pour utiliser un catalogue d'actifs, vous devez effectuer les opérations suivantes:

1. Double-cliquez sur le fichier Info.plist dans l'Explorateur de solutions pour l'ouvrir pour le modifier.
2. Faites défiler jusqu'à la section Icônes d'applications.
3. Dans la liste déroulante Source, assurez-vous que Applcons est sélectionné.
4. Dans l'Explorateur de solutions, double-cliquez sur le fichier Assets.xcassets pour l'ouvrir pour le modifier.
5. Sélectionnez Applcons dans la liste des ressources pour afficher l'éditeur d'icône.
6. Cliquez sur un type d'icône donné et sélectionnez un fichier image correspondant au type / taille requis ou faites glisser une image depuis un dossier et déposez-la à la taille souhaitée.
7. Cliquez sur le bouton Ouvrir pour inclure l'image dans le projet et définissez-la dans xcasset.

Lire Comment utiliser les catalogues d'actifs d'actifs en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6539/comment-utiliser-les-catalogues-d-actifs-d-actifs>

Chapitre 8: Connexion avec Microsoft Cognitive Services

Remarques

Dans cet exemple, nous avons utilisé le package Microsoft.ProjectOxford.Vision NuGet:

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

Pour en savoir plus sur les services cognitifs Microsoft, reportez-vous à la documentation officielle:

<https://www.microsoft.com/cognitive-services/en-us/computer-vision-api>

S'il vous plaît trouver un exemple téléchargé sur mon GitHub: https://github.com/Daniel-Krzychowski/XamarinIOS/tree/master/XamarinIOS_CognitiveServices

Je joins également un lien vers mon blog où j'ai présenté comment utiliser les services cognitifs avec l'application Xamarin Forms: <http://mobileprogrammer.pl>

Exemples

Connexion avec Microsoft Cognitive Services

Dans cet exemple, vous allez apprendre à utiliser Microsoft Cognitive Services avec l'application mobile Xamarin iOS. Nous allons utiliser l'API Computer Vision pour détecter ce qui est dans l'image.

Une fois que vous avez créé le projet Xamarin.iOS, veuillez ajouter le package NuGet ci-dessous au projet:

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

Avec cette bibliothèque, nous pourrons utiliser les services cognitifs dans notre application iOS. Je suppose que vous avez déjà un compte Microsoft enregistré pour l'utiliser et que vous avez activé Computer Vision Api comme sur l'écran ci-dessous:

Computer Vision - 5,000 transactions per month, 20 per minute.
Preview

Une fois que vous aurez cliqué sur "S'abonner", la clé Api en bas sera générée:

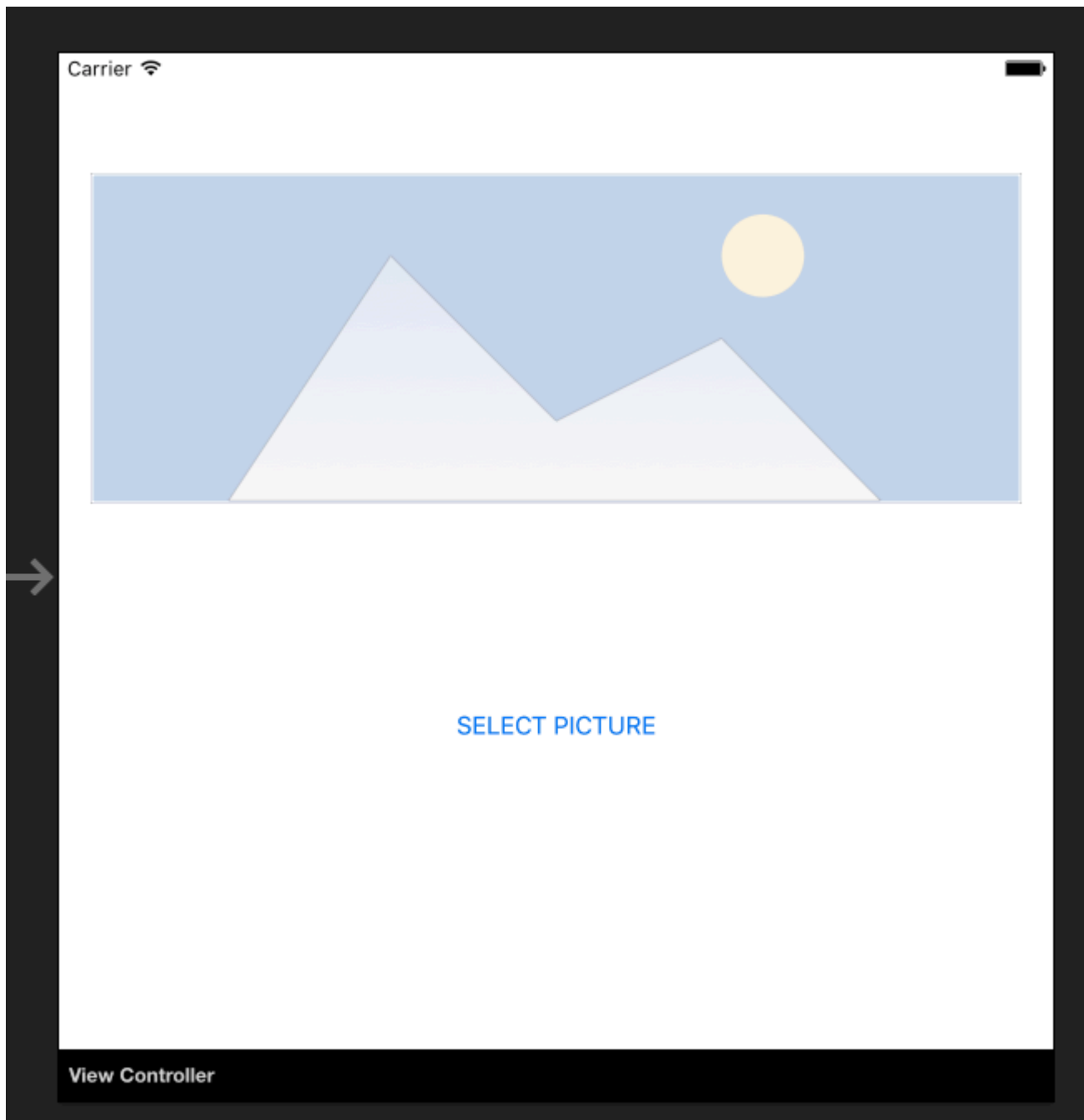
Computer Vision - Preview	5,000 transactions per month, 20 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy
---------------------------------	-------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------

Maintenant, nous pouvons commencer à configurer l'accès à Cognitive Services à partir de l'application iOS. Premièrement, nous devons avoir une image pour l'analyse. Pour ce faire, nous pouvons utiliser le composant média Xamarin disponible ci-dessous:

<https://components.xamarin.com/view/mediaplugin>

Une fois l'installation réussie, créons une interface utilisateur simple avec l'image et le bouton pour sélectionner une image dans la galerie. La taille des commandes dépend de vous.

Ouvrez Main.storyboard et ajoutez les contrôles UIImageView et UIButton par défaut ViewController. Ajoutez-leur les noms: "SelectedPictureImageView" et "SelectButton":



Maintenant, nous devons ajouter le gestionnaire d'événement "Touch Up Inside" pour gérer la sélection d'images:

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

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

Nous souhaitons maintenant afficher les informations d'analyse une fois que Cognitive Services

aura renvoyé les informations. Ajouter une étiquette sous le bouton "AnalysisLabel":



SELECT PICTURE

Analysis result....

Il est temps de connecter l'API Computer Vision!

Pour obtenir des informations sur l'image sélectionnée, ajoutez la méthode ci-dessous. N'oubliez pas de coller votre clé API!

```
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;
    }
}
```

Maintenant, vous pouvez l'invoquer dans la méthode "selectImage":

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

Une fois l'image sélectionnée, Microsoft Cognitive Services l'analysera et renverra le résultat:



SELECT PICTURE

car

Rappelez-vous que votre image ne peut pas être trop grande - dans ce cas, vous recevrez des informations comme ci-dessous:



SELECT PICTURE

Input image is too large.

Il existe de nombreux autres services que vous pouvez essayer d'utiliser. Veuillez vous référer à la documentation officielle (lien ci-joint) pour en savoir plus.

Lire Connexion avec Microsoft Cognitive Services en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6122/connexion-avec-microsoft-cognitive-services>

Chapitre 9: Contrôle de la capture d'écran dans le commutateur multitâche iOS

Introduction

Dans le [Guide de programmation d'applications pour iOS](#) :

Supprimez les informations sensibles des vues avant de passer à l'arrière-plan.

Lorsqu'une application passe en arrière-plan, le système prend un instantané de la fenêtre principale de l'application, qu'elle présente ensuite brièvement lors du passage de votre application au premier plan.

Remarques

Adapté de la question StackOverflow réelle [Contrôle de la capture d'écran dans le commutateur multitâche iOS7](#) et réponse [Obj-c](#)

Exemples

Afficher une image pour l'instantané

```
public override voidDidEnterBackground(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();
}
```

Lire [Contrôle de la capture d'écran dans le commutateur multitâche iOS en ligne](#):
<https://riptutorial.com/fr/xamarin-ios/topic/8681/controle-de-la-capture-d-ecran-dans-le-commutateur-multitache-ios>

Chapitre 10: Créer et utiliser des cellules de tableau prototype personnalisées dans xamarin.iOS à l'aide du storyboard

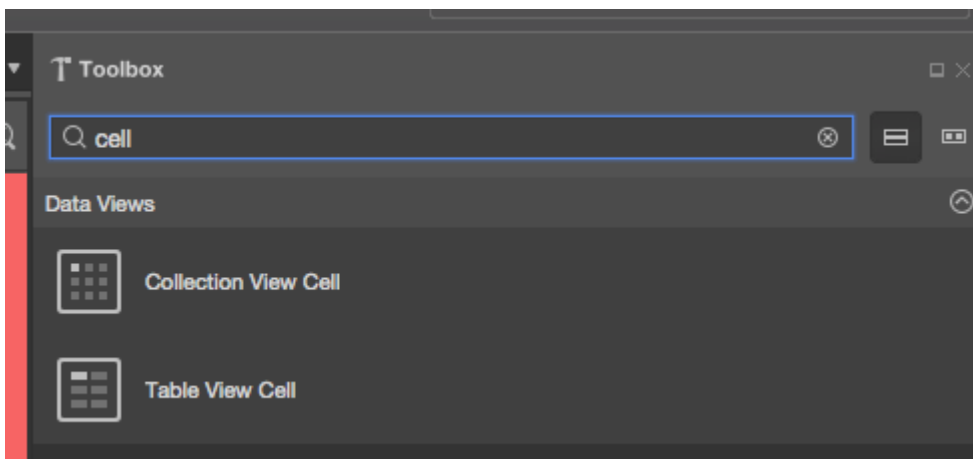
Exemples

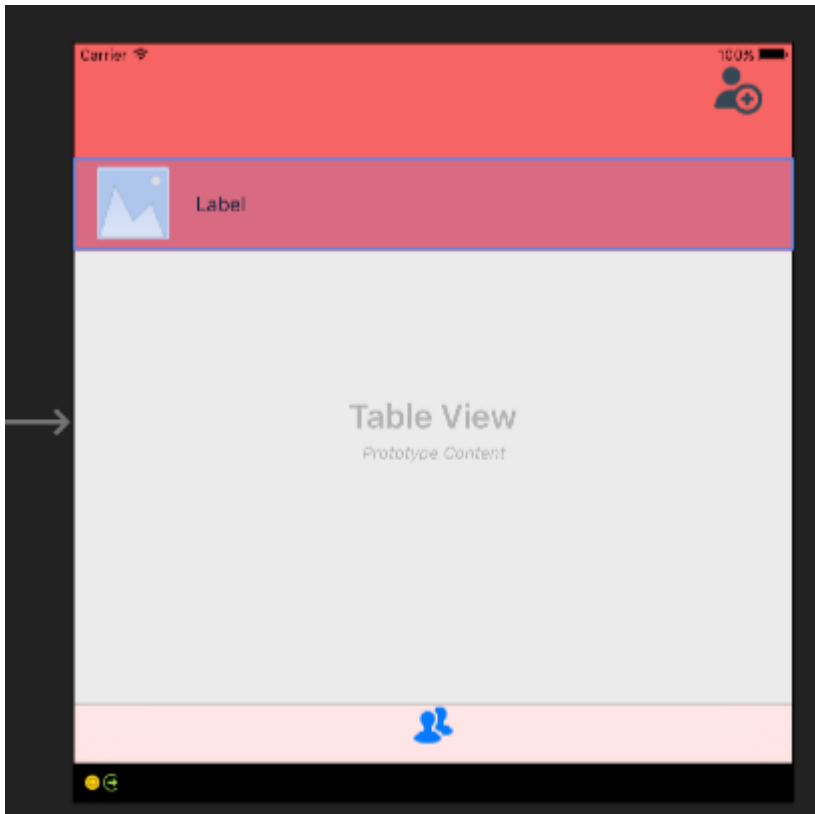
Créer une cellule personnalisée à l'aide de Storyboard

Ouvrez Storyboard où vous avez votre ViewController avec TableView:

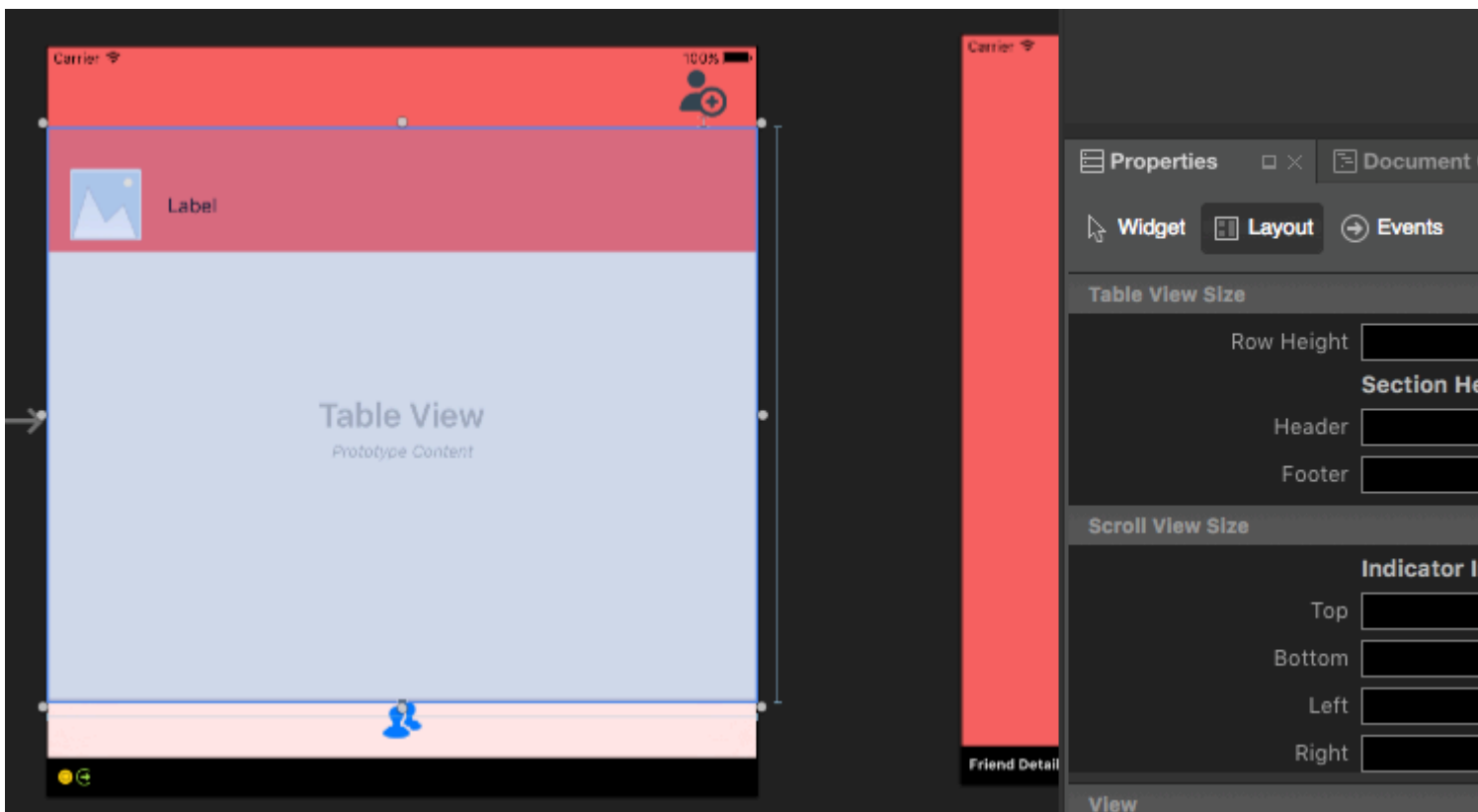
Ajouter une cellule prototype (s'il n'y a pas de cellule ajoutée auparavant):

Personnalisez la cellule comme vous le souhaitez (dans mon cas, il y a UIImage et Label personnalisés):



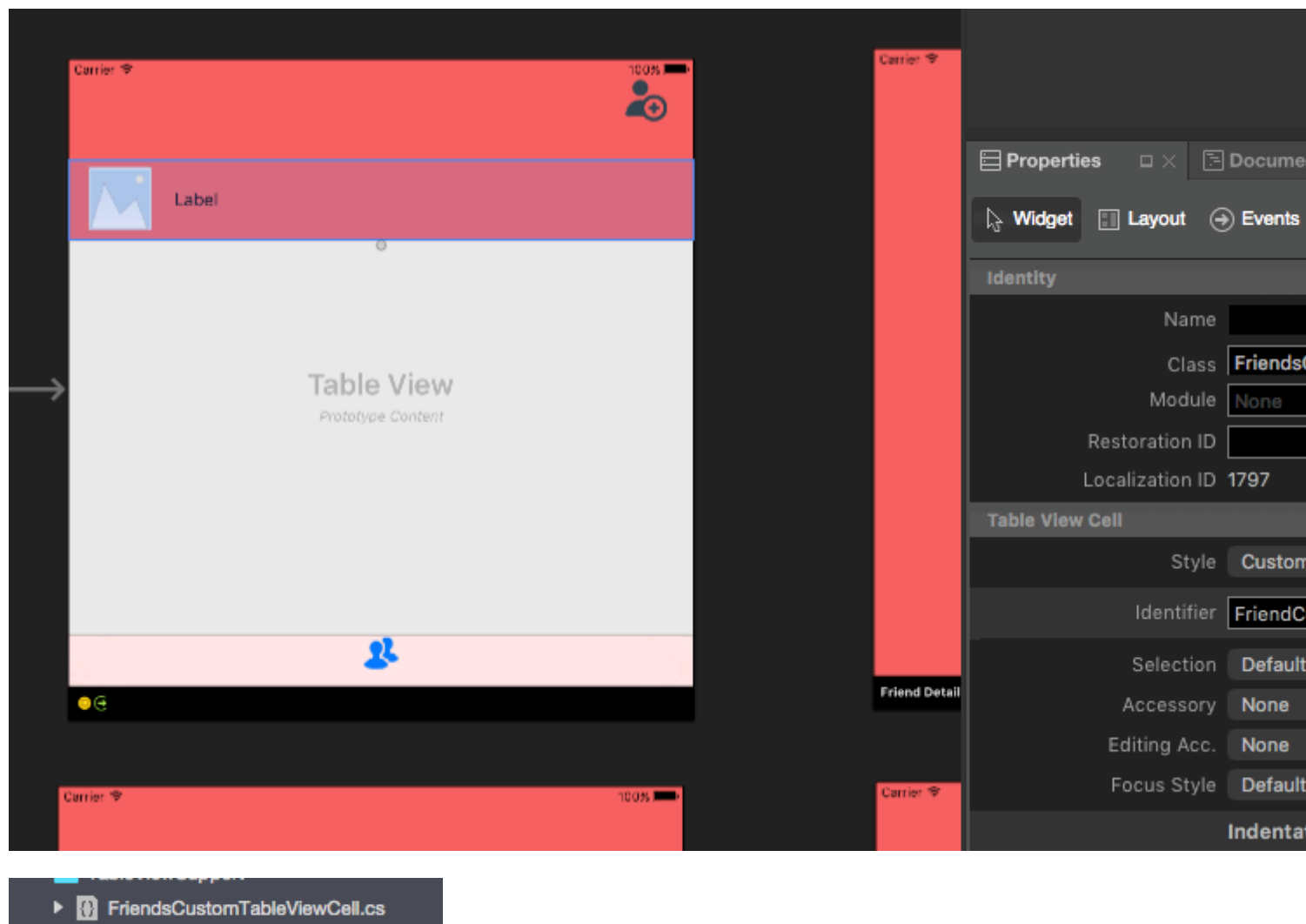


N'oubliez pas de régler la hauteur de la cellule. Pour ce faire, sélectionnez l'intégralité de TableView et, dans la fenêtre Propriétés, sélectionnez l'onglet "Mise en page". En haut de la fenêtre des propriétés, vous devriez voir "Hauteur de ligne" - mettez la valeur appropriée:



Maintenant, sélectionnez à nouveau la cellule prototype. Dans la fenêtre Propriétés, tapez le nom de la classe (elle créera une classe code-behind pour elle). Dans mon cas, il s'agit de "FriendsCustomTableViewCell". Après cela, indiquez "Identifiant" pour votre cellule. Comme vous

pouvez voir mon est "FriendCell". La dernière chose à définir est la propriété "Style" définie sur custom. Le champ "Nom" doit être vide. Une fois que vous cliquez sur "enter" après avoir tapé le fichier code-behind "Class", celui-ci sera automatiquement créé:



Maintenant, le code derrière pour la cellule devrait ressembler à ceci:

```
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;
    }
}
```

Dans UITableViewSource, vous devez déclarer cellIdentifier en haut de la classe (dans mon cas, il s'agit de "FriendCell") et dans la méthode "GetCell", vous devez convertir les cellules et définir les données correspondantes:

```
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;
}
```

Lire Créer et utiliser des cellules de tableau prototype personnalisées dans xamarin.iOS à l'aide du storyboard en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/5907/creer-et-utiliser-des-cellules-de-tableau-prototype-personnalisees-dans-xamarin-ios-a-l-aide-du-storyboard>

Chapitre 11: Des alertes

Exemples

Afficher une alerte

Pour les alertes depuis iOS 8, vous utiliseriez un `UIAlertController` mais pour les versions précédentes, vous auriez utilisé un `UIAlertView`, qui est maintenant obsolète.

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 ();
```

Afficher une alerte de connexion

Le code suivant concerne iOS 8 et versions inférieures pour créer une alerte de connexion.

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

// Setting the UIAlertViewStyle 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();

```

Afficher une feuille d'action

`UIAlertController` disponible depuis iOS8 vous permet d'utiliser le même objet d'alerte pour les feuilles d'action ou d'autres alertes classiques. La seule différence réside dans le passage de `UIAlertControllerStyle` tant que paramètre lors de la création.

Cette ligne passe d'une `UIAlertView` à une `ActionSheet`, comparée à d'autres exemples disponibles [ici](#):

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

La façon dont vous ajoutez des actions au contrôleur est toujours la même:

```

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

//Add additional actions if necessary

```

Notez que si vous avez une méthode vide sans paramètre, vous pouvez l'utiliser comme dernier paramètre de `.AddAction()`.

Par exemple, supposons que le code de `private void DoStuff(){...}` soit exécuté lorsque j'appuie sur "OK":

```

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

```

Notez que je n'utilise pas le `()` après `DoStuff` dans la création de l'action.

La façon dont vous présentez le contrôleur se fait de la même manière que tout autre contrôleur:

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

Afficher le dialogue d'alerte modale

Il était courant d'utiliser `NSRunLoop` pour afficher `UIAlertView modal UIAlertView` de bloquer l'exécution du code jusqu'à ce que l'entrée utilisateur soit traitée dans iOS; jusqu'à ce qu'Apple

publie l'iOS7, il a brisé quelques applications existantes. Heureusement, il existe une meilleure façon de l'implémenter avec l'async / wait de C #.

Voici le nouveau code tirant parti du motif asynchrone / en attente pour afficher UIAlertView modal:

```
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
}
```

Lire Des alertes en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/433/des-alertes>

Chapitre 12: ID tactile

Paramètres

Colonne	Colonne
Cellule	Cellule

Remarques

Tout d'abord, déterminez si le périphérique est capable d'accepter une entrée Touch ID.

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

Si c'est le cas, nous pouvons afficher l'interface utilisateur Touch ID en utilisant:

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

Il y a trois informations à transmettre à `EvaluatePolicy` : la politique elle-même, une chaîne expliquant pourquoi l'authentification est nécessaire et un gestionnaire de réponse. Le gestionnaire de réponse indique à l'application ce qu'il doit faire en cas de réussite ou d'échec de l'authentification.

L'une des mises en garde de l'authentification locale est qu'il doit être exécuté au premier plan. Veillez donc à utiliser `InvokeOnMainThread` pour le gestionnaire de réponse:

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

Pour déterminer si la base de données des empreintes digitales autorisées a été modifiée, vous pouvez vérifier la structure opaque (NSData) renvoyée par `context.EvaluatedPolicyDomainState`. Votre application devra stocker et comparer l'état de la stratégie pour détecter les modifications.

Une chose à noter: Apple déclare:

Cependant, la nature du changement ne peut pas être déterminée à partir de ces données.

```
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);
};
```

Exemples

Ajouter un identifiant tactile à votre application

Tout d'abord, déterminez si le périphérique est capable d'accepter une entrée Touch ID.

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

Si c'est le cas, nous pouvons afficher l'interface utilisateur Touch ID en utilisant:

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

Il y a trois informations à transmettre à `EvaluatePolicy` : la politique elle-même, une chaîne expliquant pourquoi l'authentification est nécessaire et un gestionnaire de réponse. Le gestionnaire de réponse indique à l'application ce qu'il doit faire en cas de réussite ou d'échec de l'authentification.

L'une des mises en garde de l'authentification locale est qu'il doit être exécuté au premier plan. Veillez donc à utiliser `InvokeOnMainThread` pour le gestionnaire de réponse:

```

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

```

Pour déterminer si la base de données des empreintes digitales autorisées a été modifiée, vous pouvez vérifier la structure opaque (NSData) renvoyée par `context.EvaluatedPolicyDomainState`. Votre application devra stocker et comparer l'état de la stratégie pour détecter les modifications. Une chose à noter: Apple déclare:

Cependant, la nature du changement ne peut pas être déterminée à partir de ces données.

```

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);
};

```

Exemple de bouton

```

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
};
}

```

Utilisation du trousseau

Source de travail - <https://github.com/benhysell/V.TouchIdExample>

Description du formulaire long - <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;
    }
}

```

```
}
```

Lire ID tactile en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/577/id-tactile>

Chapitre 13: Méthodes conseillées pour la migration d'UILocalNotification vers le cadre des notifications utilisateur

Exemples

UserNotifications

1. Vous devrez importer UserNotifications

```
@import UserNotifications;
```

2. Demande d'autorisation pour localNotification

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

3. Maintenant, nous mettrons à jour le numéro de badge de l'icône de l'application

```
@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()
}
```

Lire Méthodes conseillées pour la migration d'UILocalNotification vers le cadre des notifications utilisateur en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6382/methodes-conseillees-pour-la-migration-d-uilocalnotification-vers-le-cadre-des-notifications-utilisateur>

Chapitre 14: Méthodes de redimensionnement pour UIImage

Exemples

Redimensionner l'image - avec le rapport d'aspect

```
// 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;
}
```

Redimensionner l'image - sans rapport d'aspect

```
// 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;
}
```

Recadrer l'image sans redimensionner

```
// 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;  
}
```

Lire Méthodes de redimensionnement pour UIImage en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6542/methodes-de-redimensionnement-pour-uiimage>

Chapitre 15: Mise en page automatique dans Xamarin.iOS

Exemples

Ajout de contraintes avec iOS 9+ Layout Anchors

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.ConstraintGreaterThanOrEqual(BottomLayoutGuide.GetTopAnchor(), -10),
    }
);
```

Ajout de contraintes à l'aide du langage 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);
```

Vous pouvez constater que certains types de contraintes, tels que les [rapports de forme](#), ne peuvent pas être acheminés en syntaxe VFL (Visual Format Language) et doivent appeler

directement les méthodes appropriées.

Utilisation de Cirrious.FluentLayout

Utiliser NuGet

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

Un exemple développé basé sur l'exemple de démarrage de la page [GitHub](#) , un simple prénom, des étiquettes de noms et des champs empilés les uns sur les autres:

```
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));
}
```

Ajout de contraintes avec la maçonnerie

La maçonnerie est une bibliothèque d'objectifs-c mais xamarin a créé une liaison pour elle et l'a créée comme un paquet nuget <https://www.nuget.org/packages/Masonry/> .

Nuget installer

```
Install-Package Masonry
```

Cela centre un bouton de 100 points sous le point central de la vue contenant et définit une largeur comprise entre 200 et 400 points

```
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));
});
```

Cela définit une image à l'échelle 100 points au-dessus du point central de la vue contenant puis définit la largeur à la largeur de la vue contenant un multiplicateur de 0,5, ce qui signifie 50% de la largeur. Il définit ensuite la hauteur à la largeur multipliée par le ratio d'aspect qui entraîne la mise à l'échelle de l'image mais conserve son rapport d'aspect correct

```
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);
});
```

Lire Mise en page automatique dans Xamarin.iOS en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/1317/mise-en-page-automatique-dans-xamarin-ios>

Chapitre 16: Programmation concurrente dans Xamarin.iOS

Exemples

Manipulation de l'interface utilisateur à partir de threads d'arrière-plan

Les threads d'arrière-plan ne peuvent pas modifier l'interface utilisateur. presque toutes les méthodes UIKit doivent être appelées sur le thread principal.

A partir d'une sous-classe de `NSObject` (y compris `UIViewController` ou `UIView`):

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

A partir d'une classe C # standard:

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

`InvokeOnMainThread` attend l' `InvokeOnMainThread` de votre code sur le thread principal avant de continuer. Si vous n'avez pas besoin d'attendre, utilisez `BeginInvokeOnMainThread` .

Utiliser Async et attendre

Vous pouvez utiliser des méthodes asynchrones pour gérer les exécutions asynchrones. Par exemple, requêtes POST et GET. Laissez dire ci-dessous est votre méthode d'acquisition de données.

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

Vous pouvez appeler cette méthode comme indiqué ci-dessous

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

Cependant, dans la pratique, cette méthode sera dans une interface de couche de service. Le meilleur moyen d'y parvenir est de créer une méthode distincte pour appeler ceci et mettre à jour l'interface utilisateur illustrée ci-dessous.

```
//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();
}
```

Dans l'extrait de code ci-dessus, la méthode `GetDataListFromServer` sera appelée et enverra la requête Web. Néanmoins, il ne bloquera pas le thread d'interface utilisateur tant qu'il n'aura pas reçu la réponse du serveur. Il va descendre la ligne après `await GetDataListFromServer(1)`. Cependant, dans la méthode `private async Task GetArchivedListFromServer(int type)`, il attendra qu'il obtienne la réponse du serveur pour exécuter les lignes après que `var result = await GetDataFromServer(type)`.

Lire Programmation concurrente dans Xamarin.iOS en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/1364/programmation-concurrente-dans-xamarin-ios>

Chapitre 17: Reliure des bibliothèques rapides

Introduction

Un guide facile à suivre qui vous guidera à travers le processus de liaison des fichiers `.framework` Swift pour une utilisation dans un projet Xamarin.

Remarques

1. Lors de la construction d'une bibliothèque dans Xcode, il est possible d'inclure les bibliothèques rapides. Ne pas! Ils seront inclus dans votre application finale sous `NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib`, mais ils doivent être inclus sous `NAME.app/Frameworks/libswift*.dylib`.
2. Vous pouvez trouver cette information ailleurs, mais cela vaut la peine de le mentionner: N'incluez pas le Bitcode dans la bibliothèque. À l'heure actuelle, Xamarin n'inclut pas Bitcode pour iOS et Apple exige que toutes les bibliothèques prennent en charge les mêmes architectures.

Exemples

Relier une bibliothèque Swift à Xamarin.iOS

Reliure une bibliothèque Swift à Xamarin.iOS suit le même processus pour Objective-C comme le montre https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/ , mais avec quelques mises en garde.

1. Une classe swift doit hériter de `NSObject` pour être liée.
2. Compilateur Swift se traduira par des noms de classe et protocole en quelque chose d' autre à moins que vous utilisez l' `@objc` annotation (par exemple `@objc (MyClass)`) dans vos classes rapides pour spécifier le nom objectif explicite `c`.
3. Pendant l'exécution, votre application doit inclure des bibliothèques de base rapides parallèlement à votre infrastructure liée dans un dossier appelé `Frameworks`;
4. Lorsque l'application est transmise à AppStore, elle doit inclure un dossier `SwiftSupport` avec votre dossier `Payload`. Ceux-ci sont dans le fichier IPA.

Ici vous pouvez trouver un lien simple:

<https://github.com/Flash3001/Xamarin.BindingSwiftLibrarySample>

Et un exemple de liaison complète: <https://github.com/Flash3001/iOSCharts.Xamarin>

Veillez trouver les étapes ci-dessous:

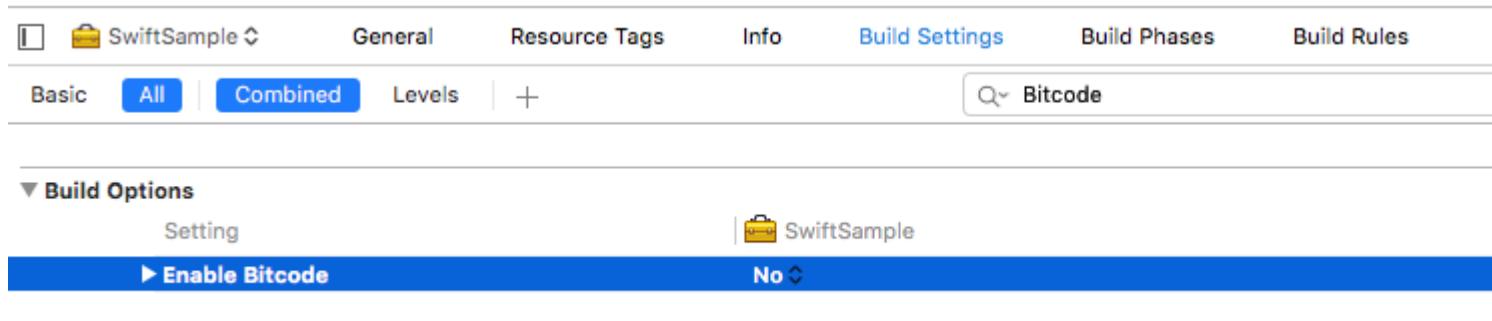
1.1 Préparez les classes Swift que vous souhaitez exporter

Pour toute classe Swift que vous souhaitez utiliser, vous devez soit hériter de NSObject et rendre le nom Objective-C explicite à l'aide de l'annotation objc. Sinon, le compilateur Swift générera des noms différents. Vous trouverez ci-dessous un exemple de code de la façon dont une classe Swift pourrait ressembler. Notez que la classe dont elle hérite n'a pas d'importance tant que la classe racine hérite de NSObject.

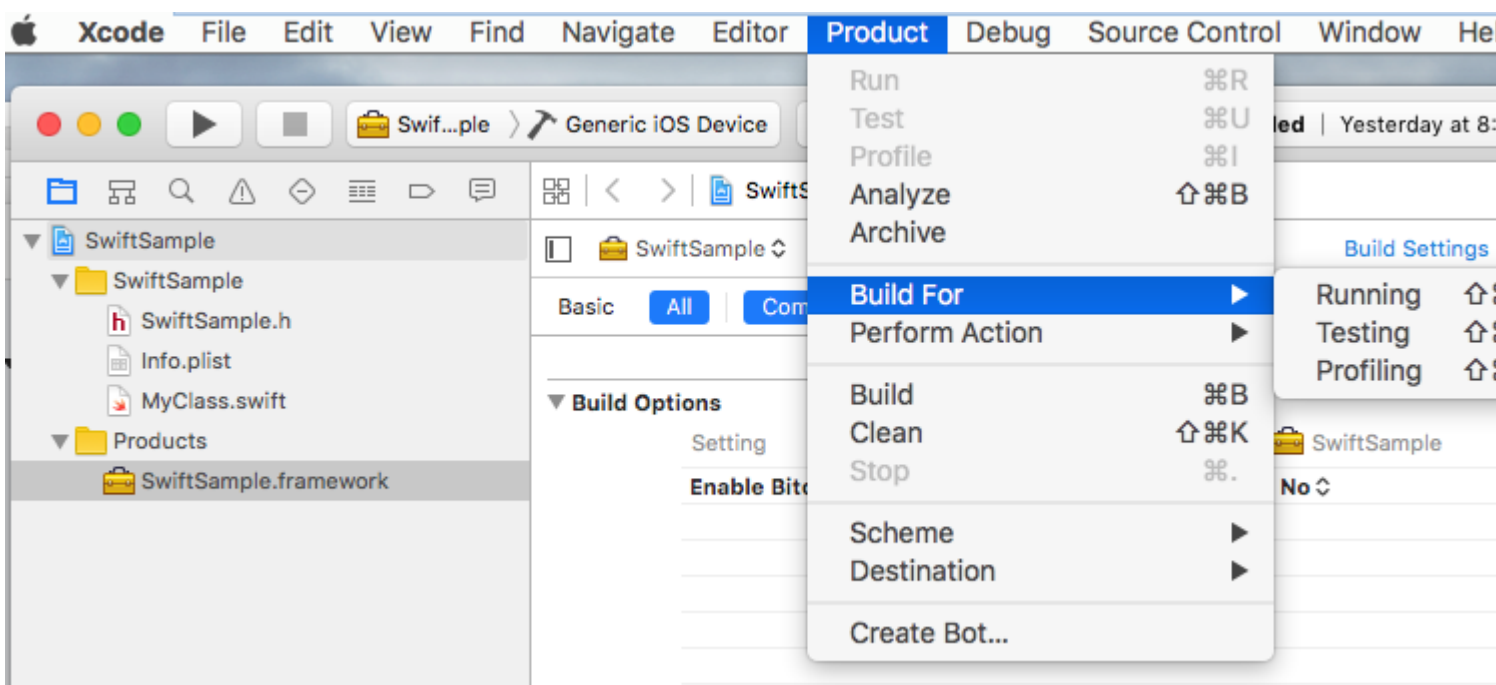
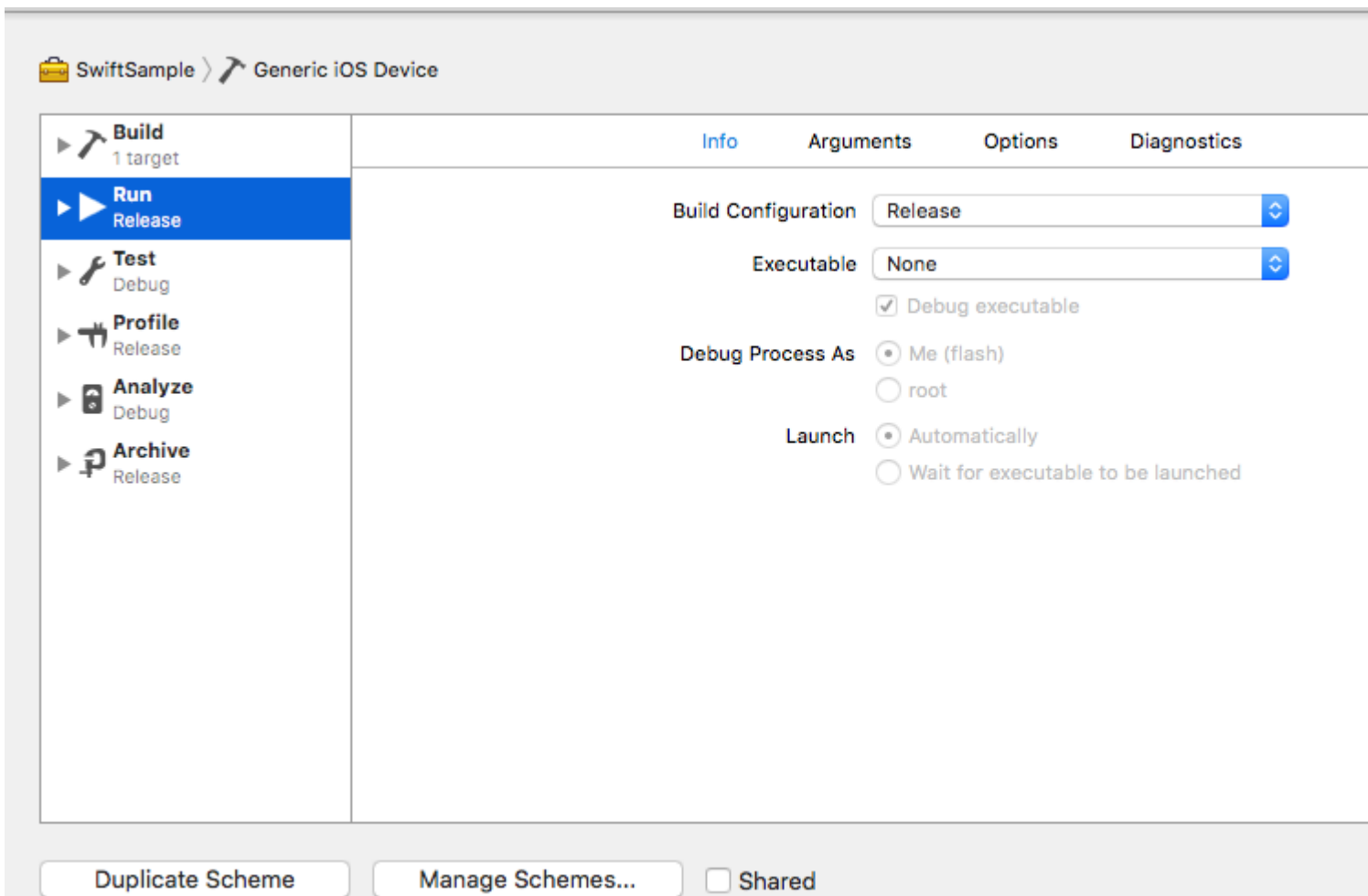
```
//Add this to specify explicit objective c name
@objc(MyClass)
open class MyClass: NSObject {
    open func getValue() -> String
    {
        return "Value came from MyClass.swift!";
    }
}
```

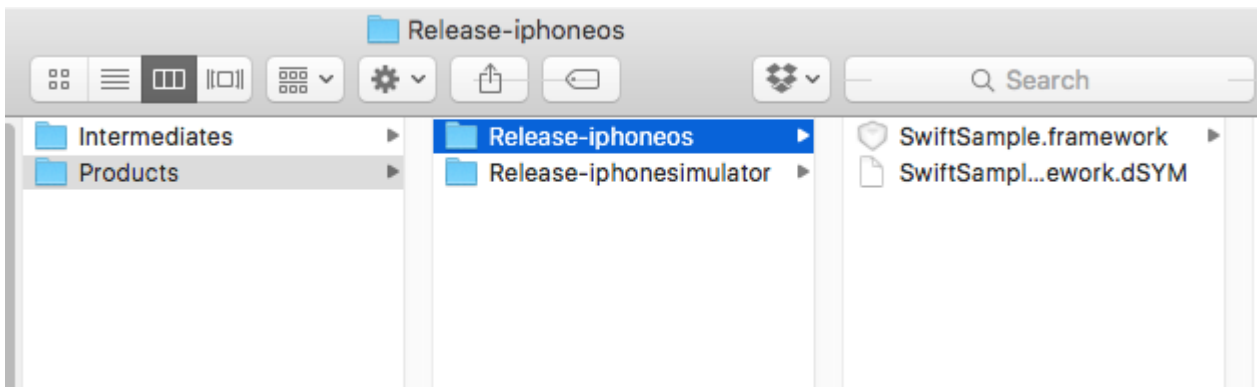
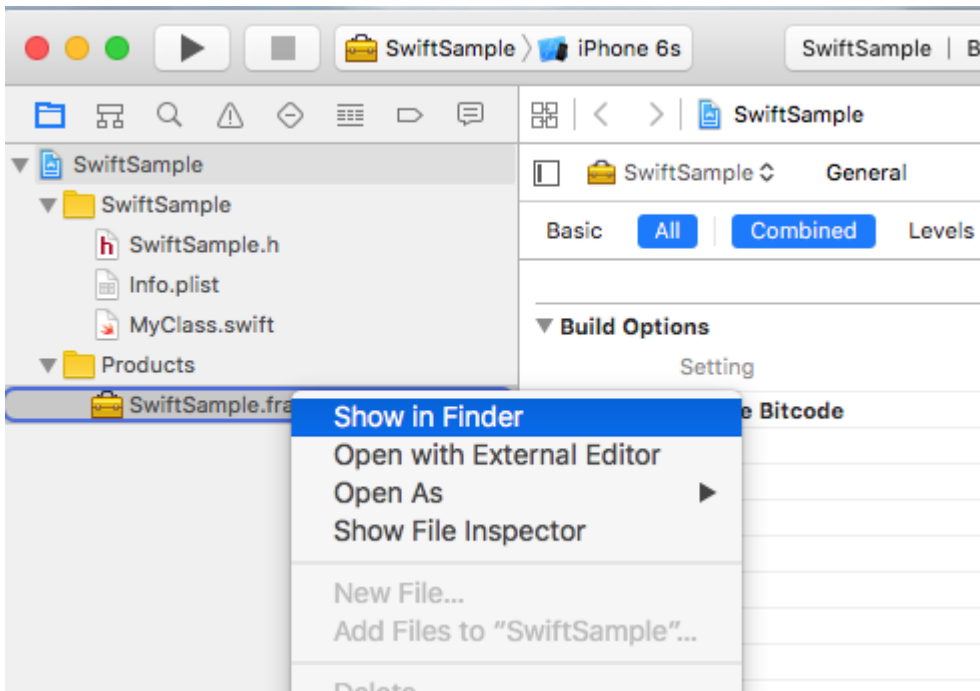
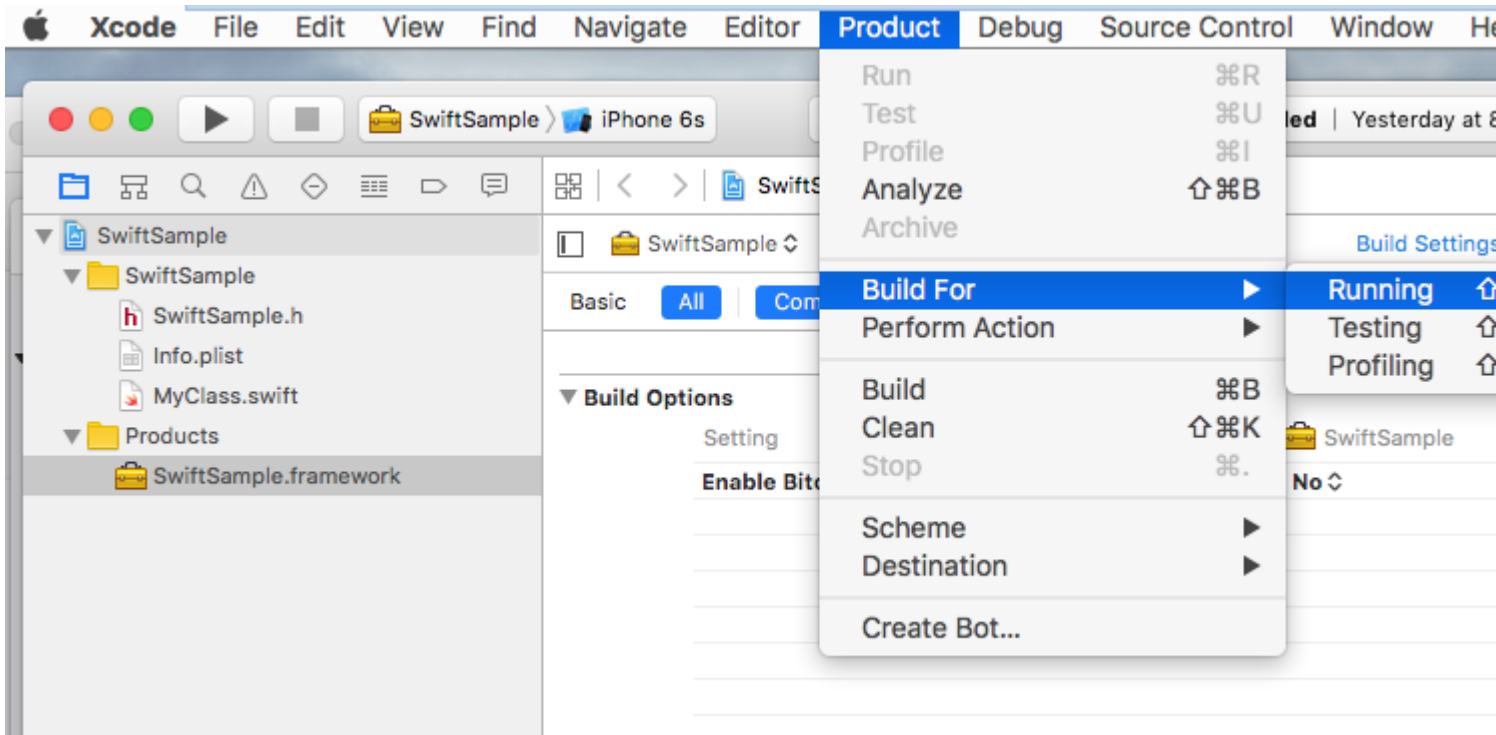
1.2 Construire le cadre

Désactiver le code binaire *



Construire pour version pour Device and Simulator. *



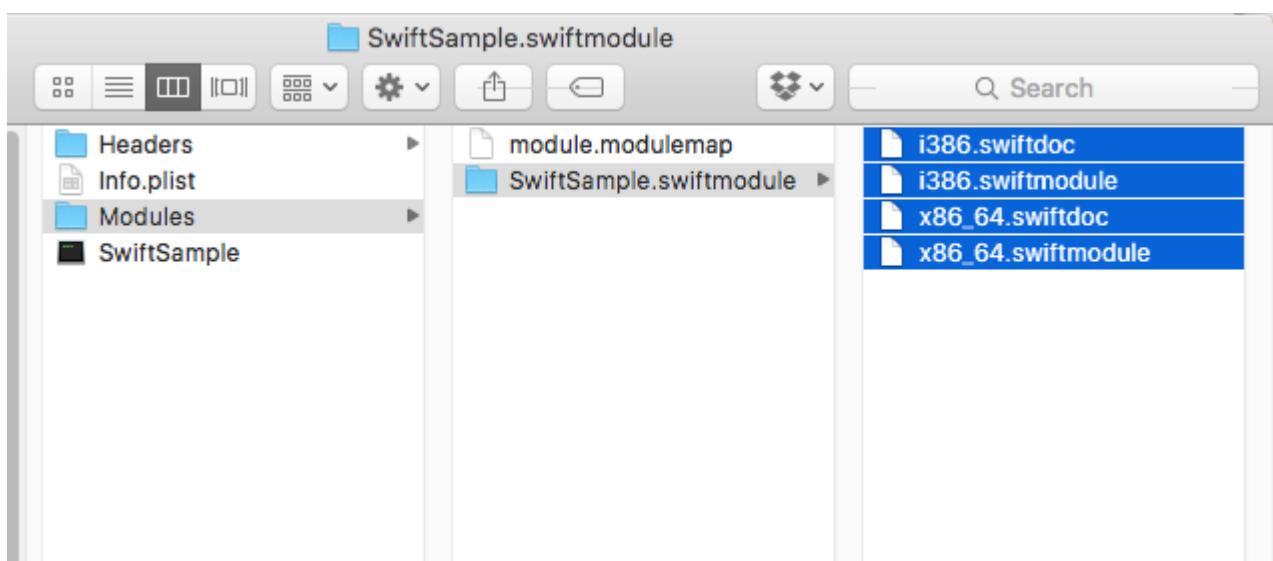
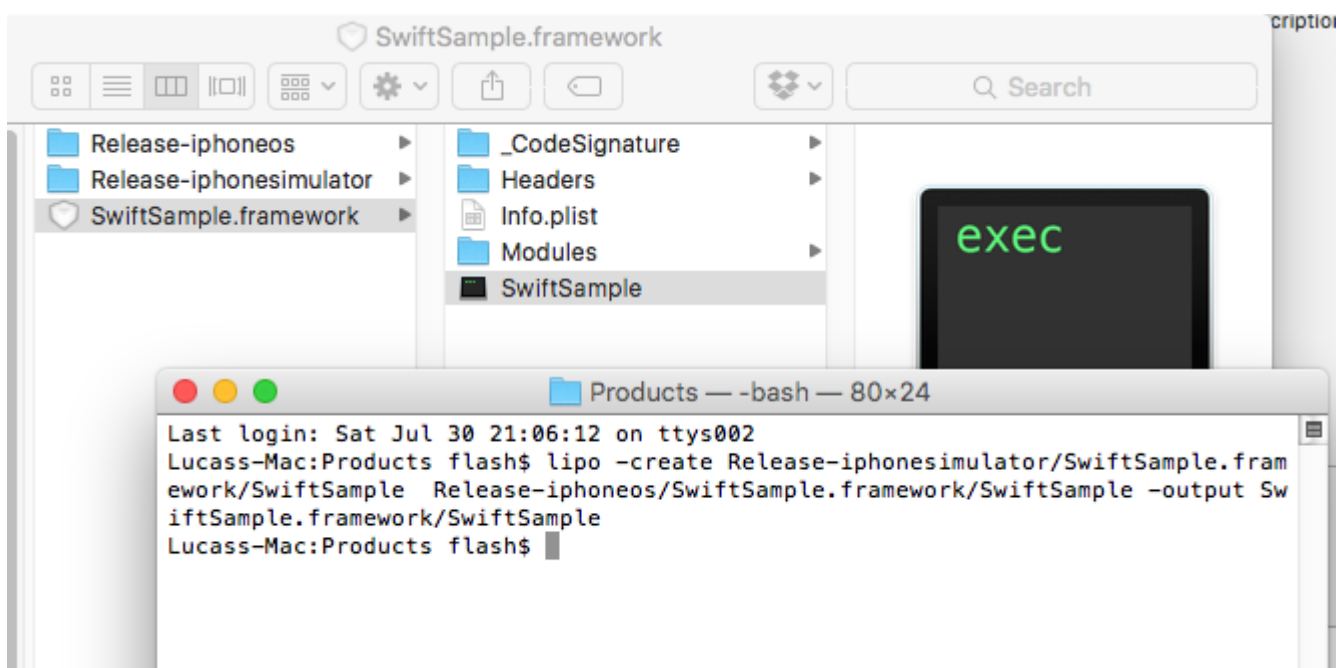


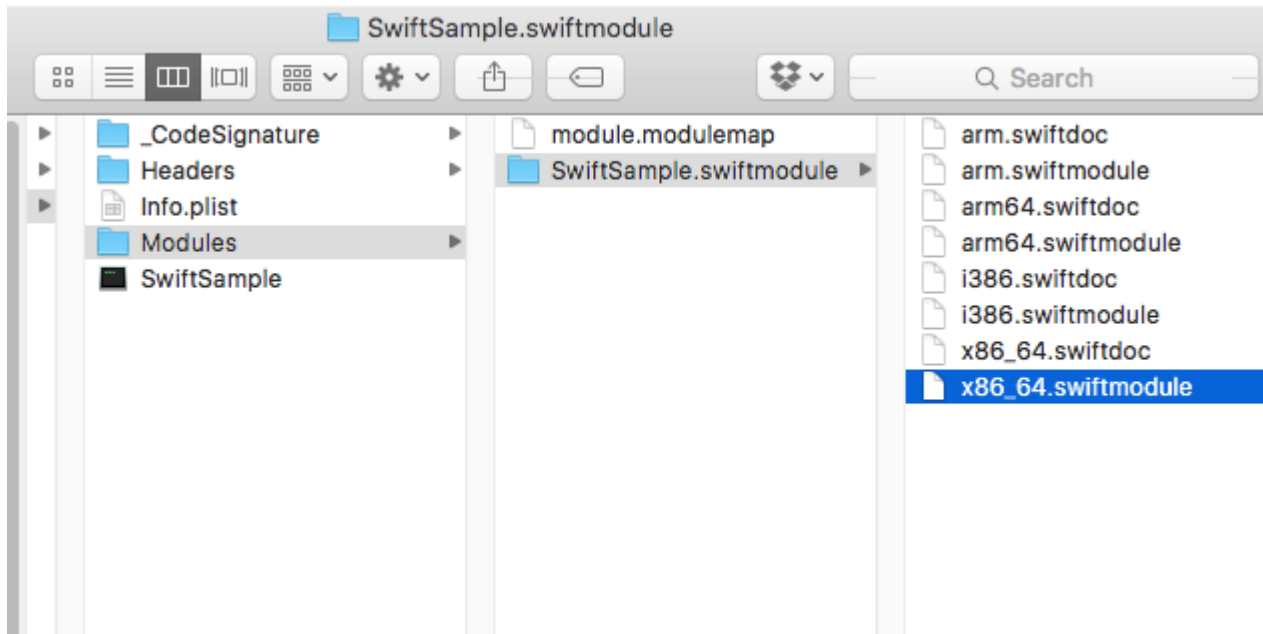
- Non lié uniquement à la liaison Swift.

2. Créez une bibliothèque de graisse

Un framework contient plusieurs fichiers, celui qui doit manger un peu est NAME.framework / NAME (sans extension).

- Copier Release-iphoneos / NAME.framework sur NAME.framework
- Créez la bibliothèque FAT en utilisant:
 - **lipo -create Release-iphonesimulator / NAME.framework / NAME Version-iphoneos / NAME.framework / NAME -output NAME.framework / NAME**
- Copiez les fichiers dans la version-iphonesimulator / NAME.framework / modules / NAME.swiftmodule à NAME.framework / modules / NAME.swiftmodule (jusqu'à présent, il ne contient que les fichiers des iphoneos)

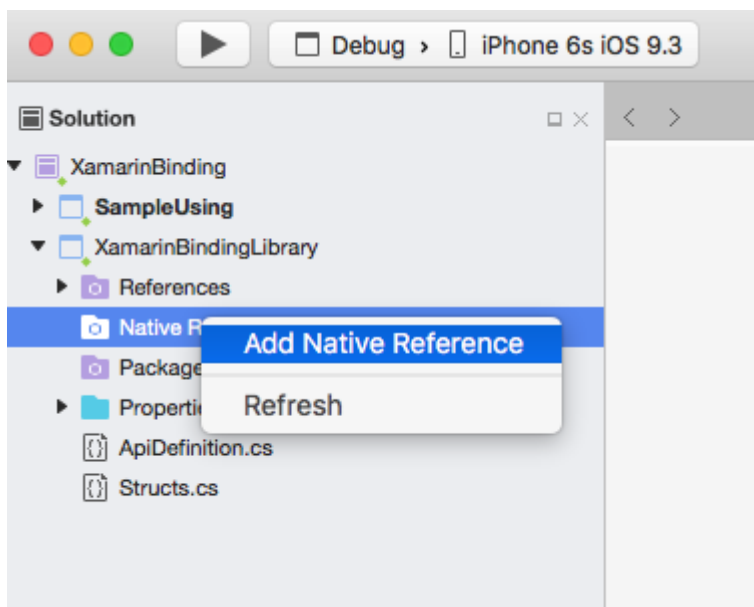


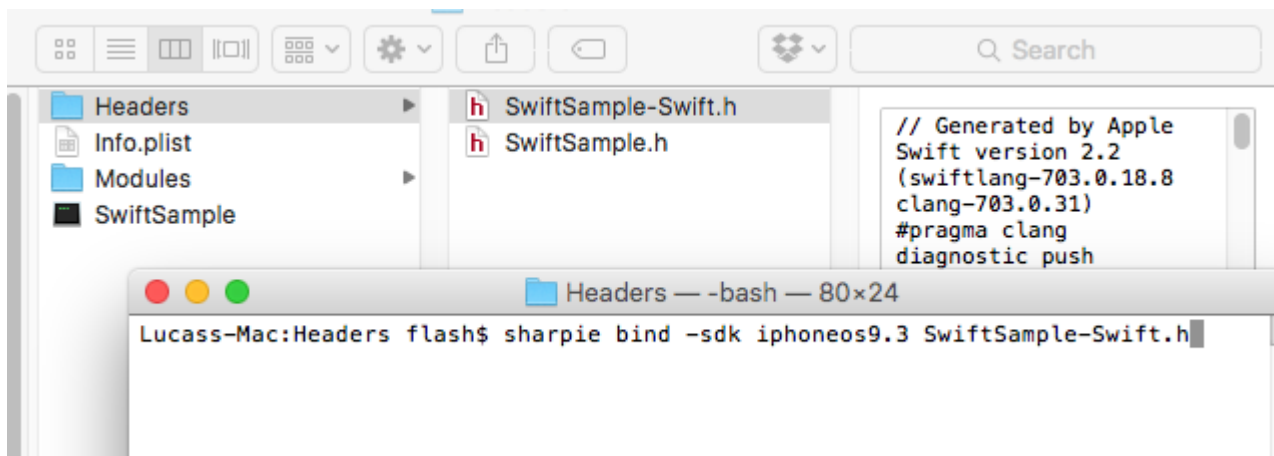
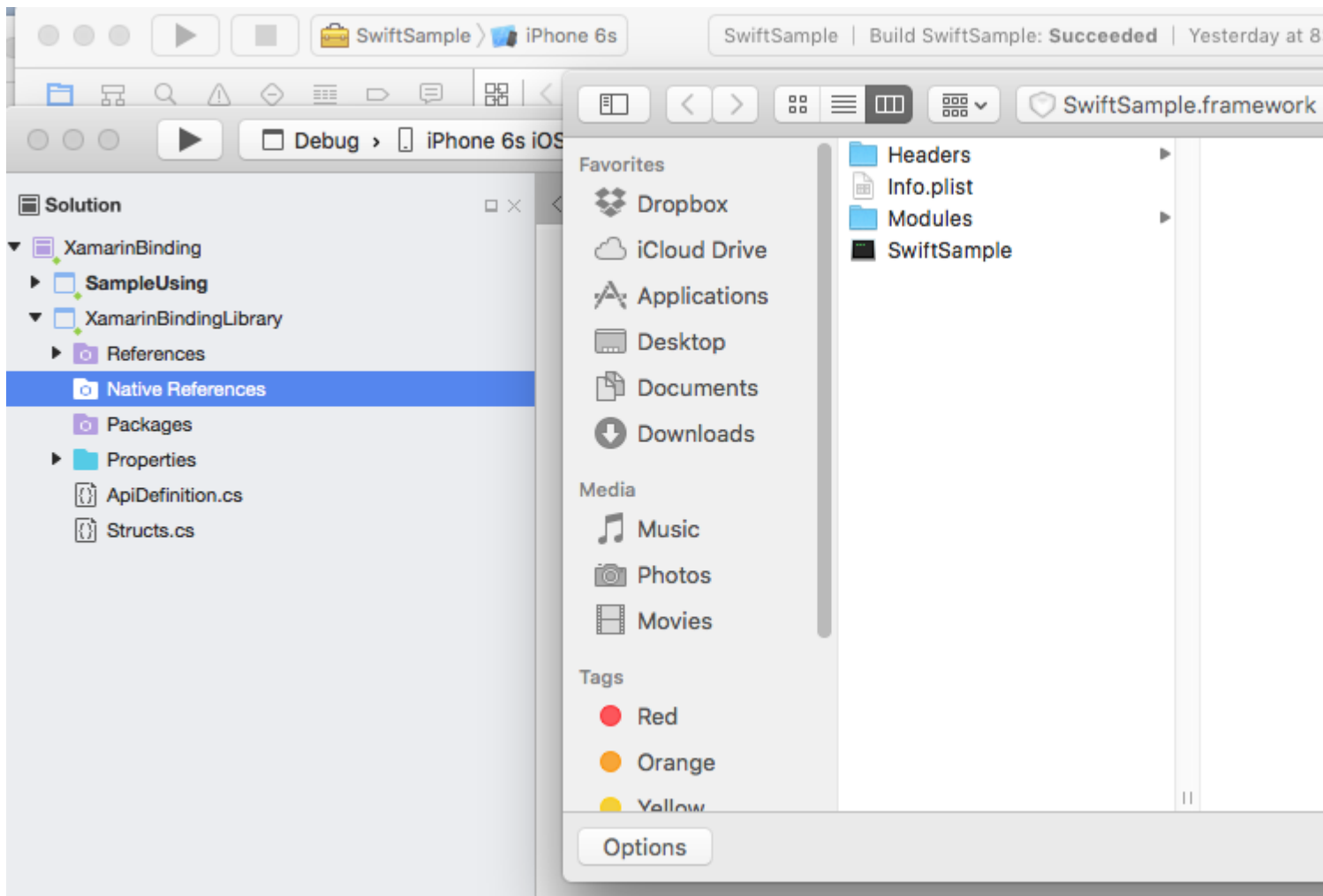


3. Importer la bibliothèque

Je suppose que vous avez déjà créé le projet Binding dans File -> New -> iOS -> Binding Library.

Xamarin prend en charge l'importation de .frameworks. Cliquez avec le bouton droit sur "Native References" et cliquez sur "Ajouter une référence native". Trouvez le nouveau cadre de travail créé et ajoutez-le.





4. Créez le fichier ApiDefinition basé sur le fichier LIBRARY-Swift.h dans les en-têtes.

Vous pouvez le faire manuellement, mais ce ne sera pas agréable. Vous pouvez utiliser l'objectif Sharpie. L'outil que Xamarin utilise pour lier ses propres bibliothèques.

Comment l'utiliser sur <https://developer.xamarin.com/guides/cross-platform/macios/binding/objective-sharpie/>

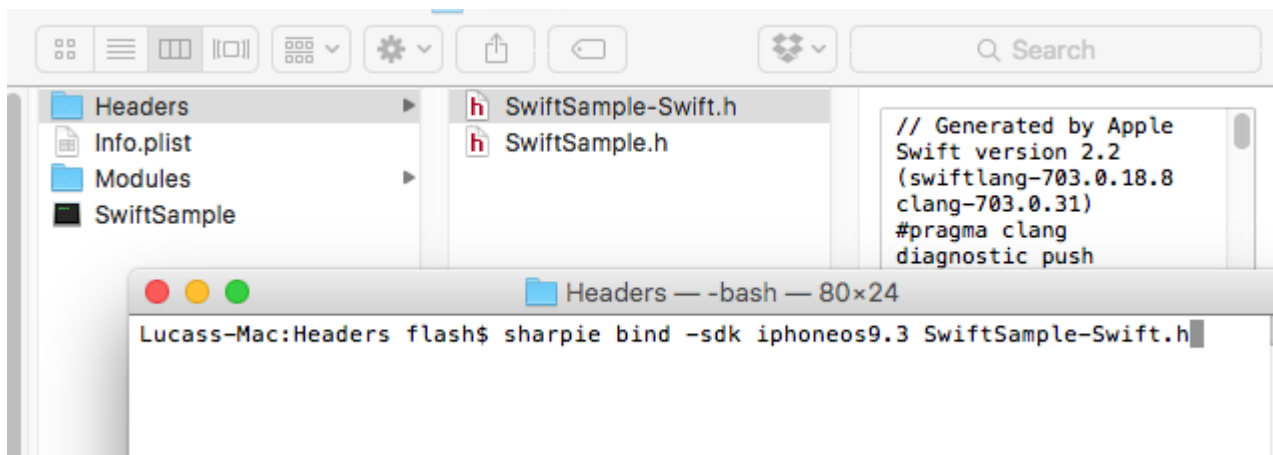
La commande de base sera quelque chose comme: **sharpie bind -sdk iphoneos9.3 NAME-Swift.h**

Si vous obtenez une `System.Reflection.TargetInvocationException` c'est probablement parce que vous avez installé une autre version du SDK. Exécutez la commande suivante pour vérifier avec iPhone OS SDK que vous avez installé:

```
sharpie xcode -sdks
```

Le fichier **NAME-Swift.h** se trouve dans **NAME.framework / Headers / NAME-Swift.h**

Note: Les classes swift doivent hériter de "NSObject", sinon **NAME-Swift.h n'importera** pas vos classes et Objective Sharpie ne convertira rien.



Remplacez le contenu de votre projet de liaison ApiDefinition.cs par celui nouvellement créé.

```
ApiDefinition.cs
MyClass ▶ No selection
1 using Foundation;
2
3 namespace XamarinBindingLibrary
4 {
5     // @interface MyClass : NSObject
6     [BaseType(typeof(NSObject))]
7     interface MyClass
8     {
9         // -(NSString * _Nonnull)getValue;
10        [Export("getValue")]
11        string Value { get; }
12    }
13 }
14
```

5. Modifiez tous les protocoles [Protocol] et [BaseType] pour inclure le nom de la classe dans le runtime Objective-C.

Dans le cas où la classe Swift d'origine ou le protocole ne comprend pas l'annotation `@objc` (`MyClass`) comme spécifié à l'étape 1.1, ils auront les noms Objective-C internes ont changé, vous devez mapper à la bonne.

Tous les noms sont disponibles dans le fichier `NAME-Swift.h` au format suivant:

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

Et

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

Pour définir le nom que vous utilisez `BaseTypeAttribute.Name`

<https://developer.xamarin.com/guides/cross-platform/macios/binding/binding-types-reference/#BaseType.Name> propriété pour les classes et `ProtocolAttribute.Name` <https://developer.xamarin.com/api/property/MonoTouch.Foundation.ProtocolAttribute.Name/> pour les protocoles.

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

Le faire manuellement n'est pas cool. Vous pouvez utiliser cet outil

<https://github.com/Flash3001/SwiftClassify> pour insérer tous les noms. (C'est du travail en cours. Mais c'est assez simple, simplement en regardant le code, vous obtiendrez comment cela fonctionne).

6.1 Inclure toutes les dépendances Swift à exécuter.

Si vous essayez de consommer la bibliothèque dans une application et que vous essayez de l'exécuter immédiatement, elle se bloquera. L'erreur est due à l'absence de `libswiftCore.dylib`

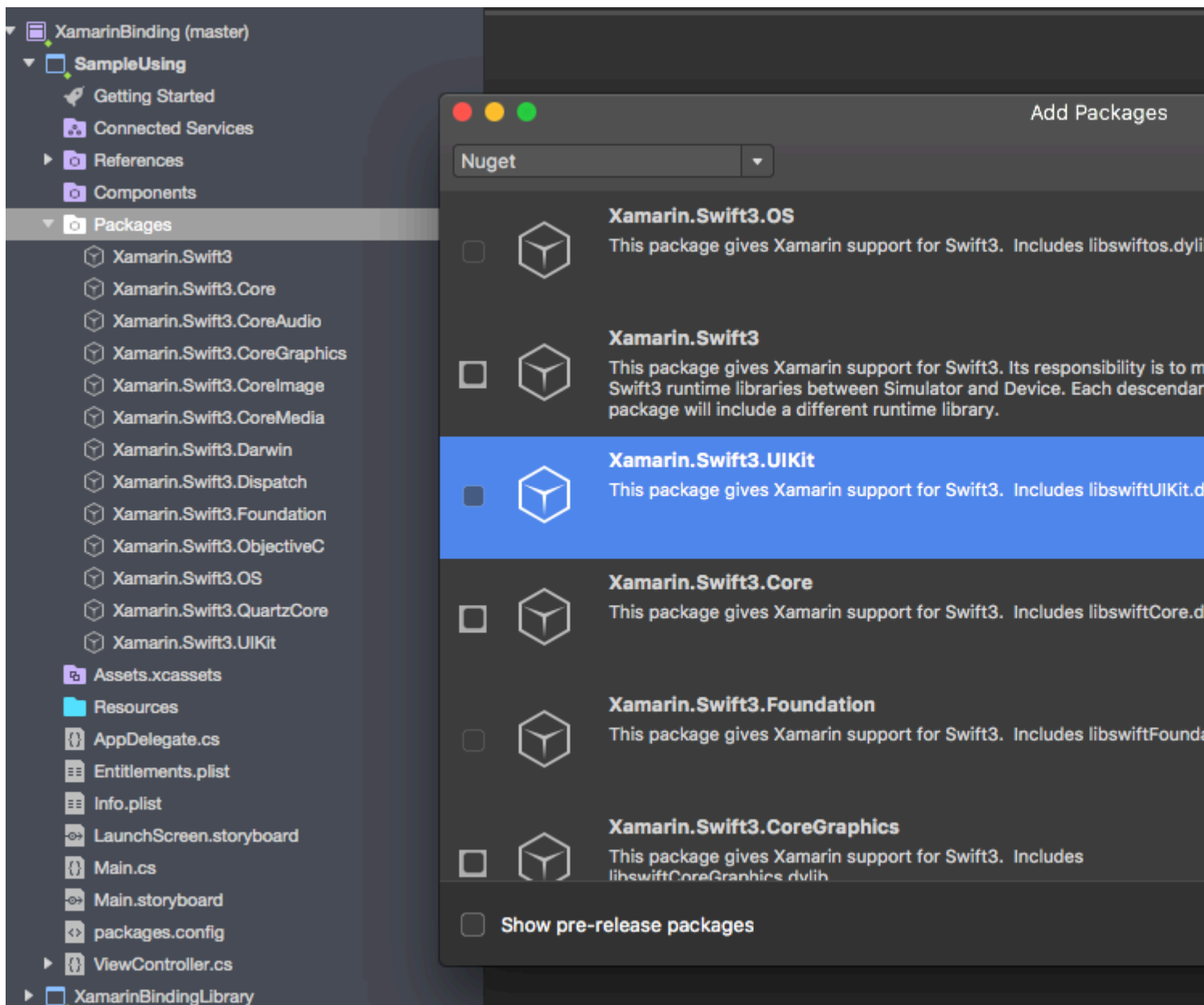
Quelque chose comme ça:

```
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 ne fournit pas de support officiel pour lier une bibliothèque Swift. Vous devez donc inclure manuellement les bibliothèques de base swift dans les dossiers `Frameworks` et `SwiftSupport`. Les fichiers du dossier `Frameworks` sont différents pour `Simulator` et `Device`. Ils peuvent être trouvés dans

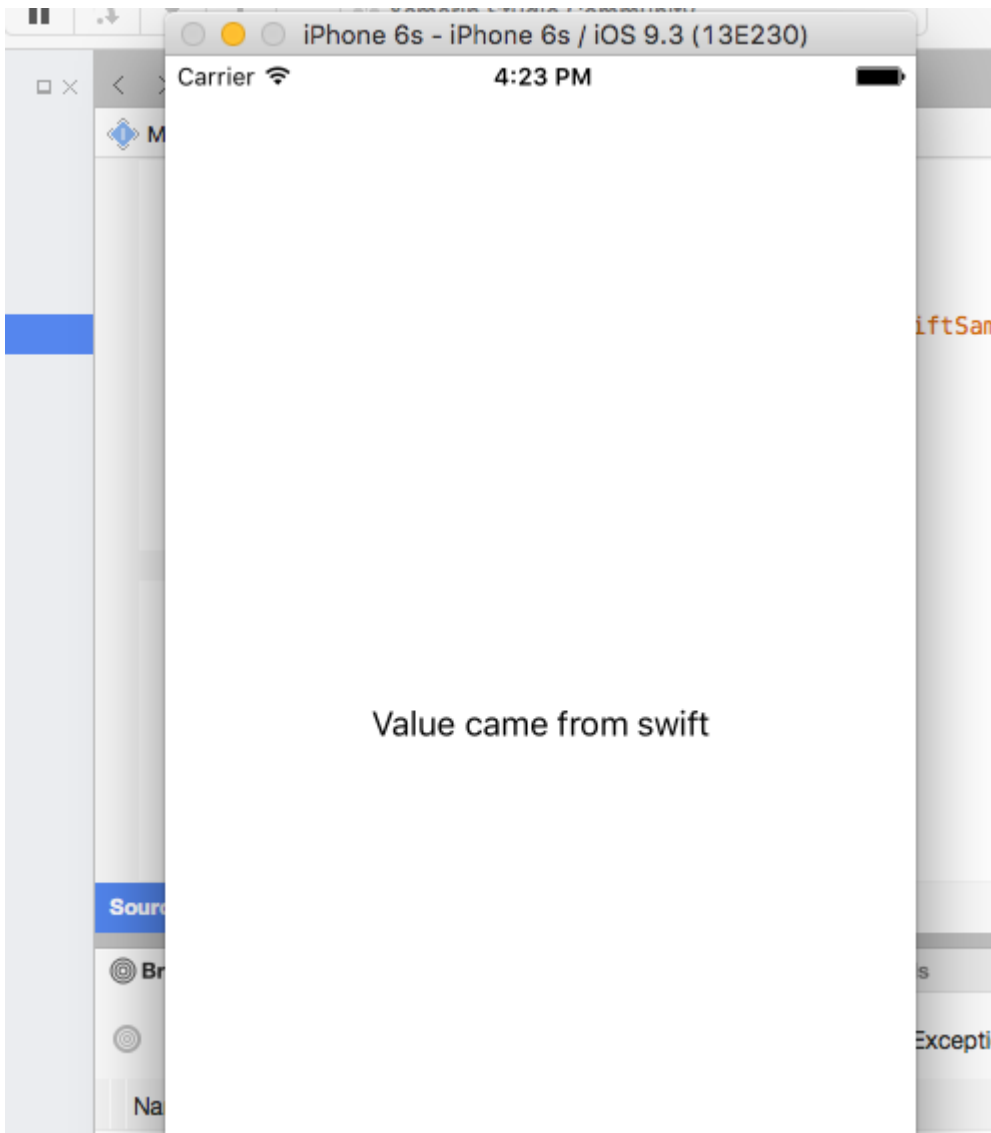
/Applications/Xcode.app/Contents/Developer//XcodeDefault.xctoolchain/usr/lib/swift.Toolchains

Au lieu de copier manuellement les fichiers dans le dossier Framework, vous pouvez utiliser cette bibliothèque <https://github.com/Flash3001/Xamarin.Swift3.Support> . Il inclut toutes les dépendances dont Swift 3.1 a besoin, chacune dans un seul package NuGet.



Comme vous pouvez le voir, le package Nuget est inclus dans l'application grand public, pas dans la liaison elle-même. Si vous essayez de l'inclure dans la liaison, vous obtiendrez des erreurs de compilation.

Si vous créez un package Nuget, vous pouvez demander à Nuget de l'inclure en tant que dépendance.



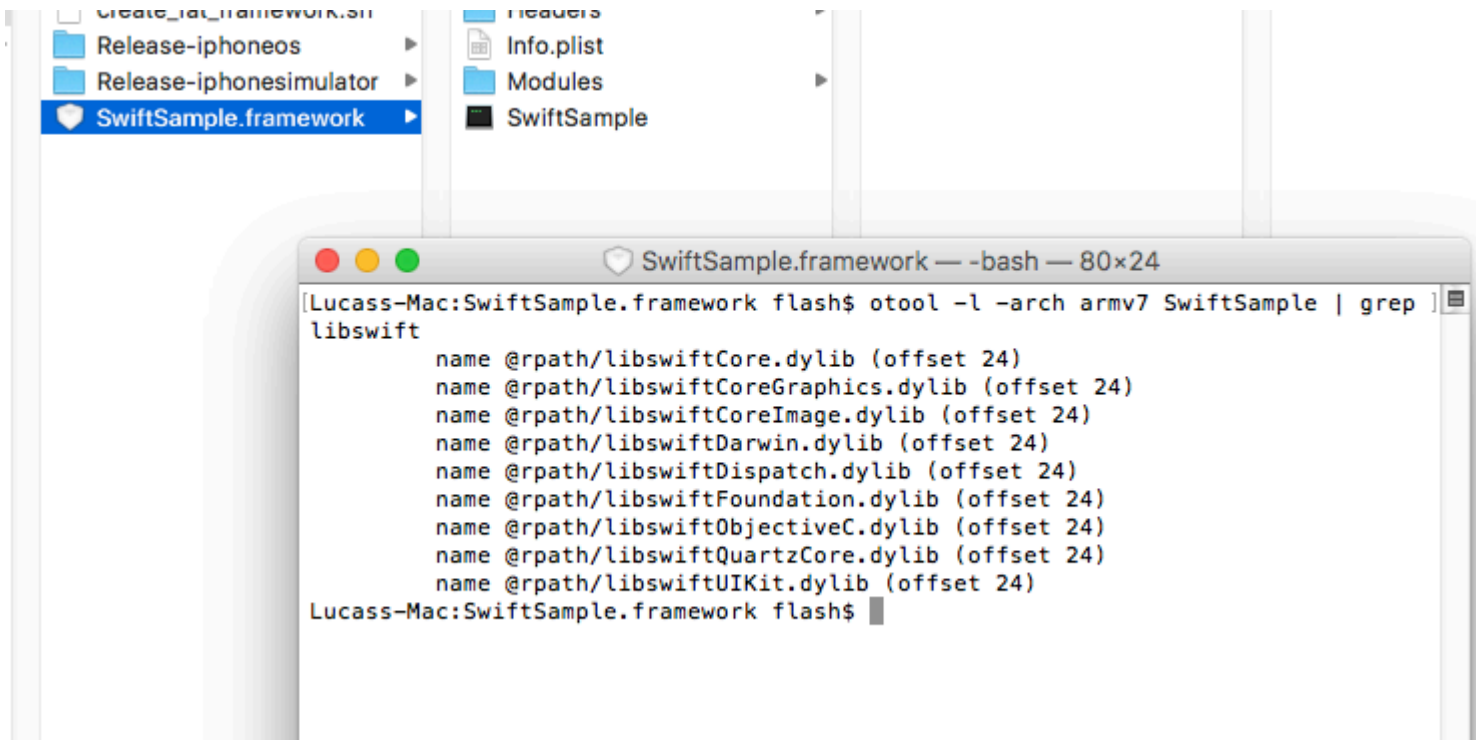
6.2. Identifier les dépendances Swift à inclure.

Une chose importante à faire est de déterminer chaque paquet que vous devez inclure dans votre projet. Une liaison simple nécessitera généralement:

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

Pour répertorier chaque dépendance, vous pouvez exécuter la commande suivante dans votre `LibraryName.framework`

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



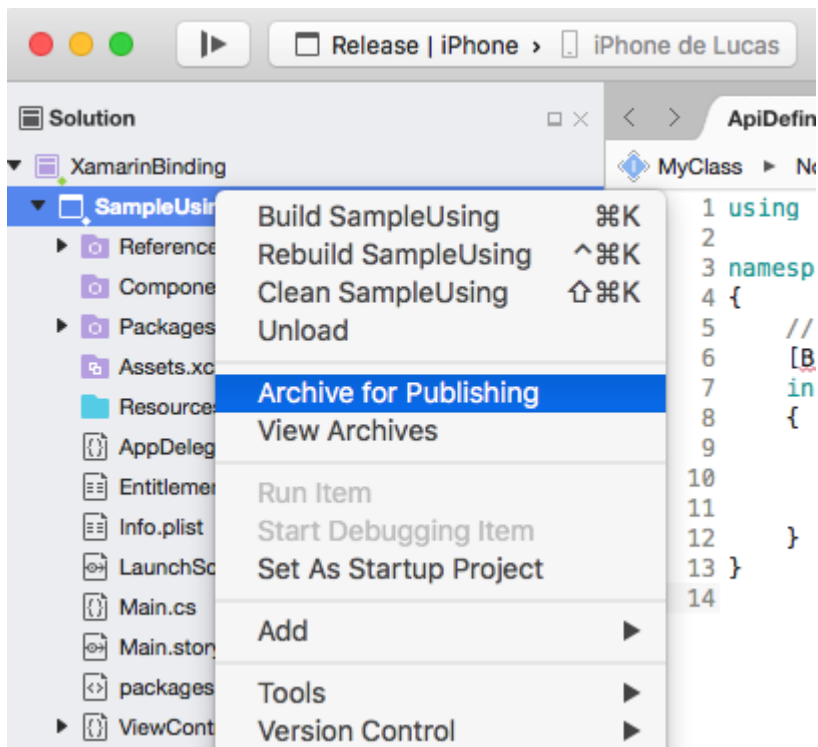
N'incluez pas tous les packages disponibles dans NuGet for Swift3, car ils pourraient augmenter la taille de votre application.

7. Incluez SwiftSupport pour pousser l'application vers AppStore.

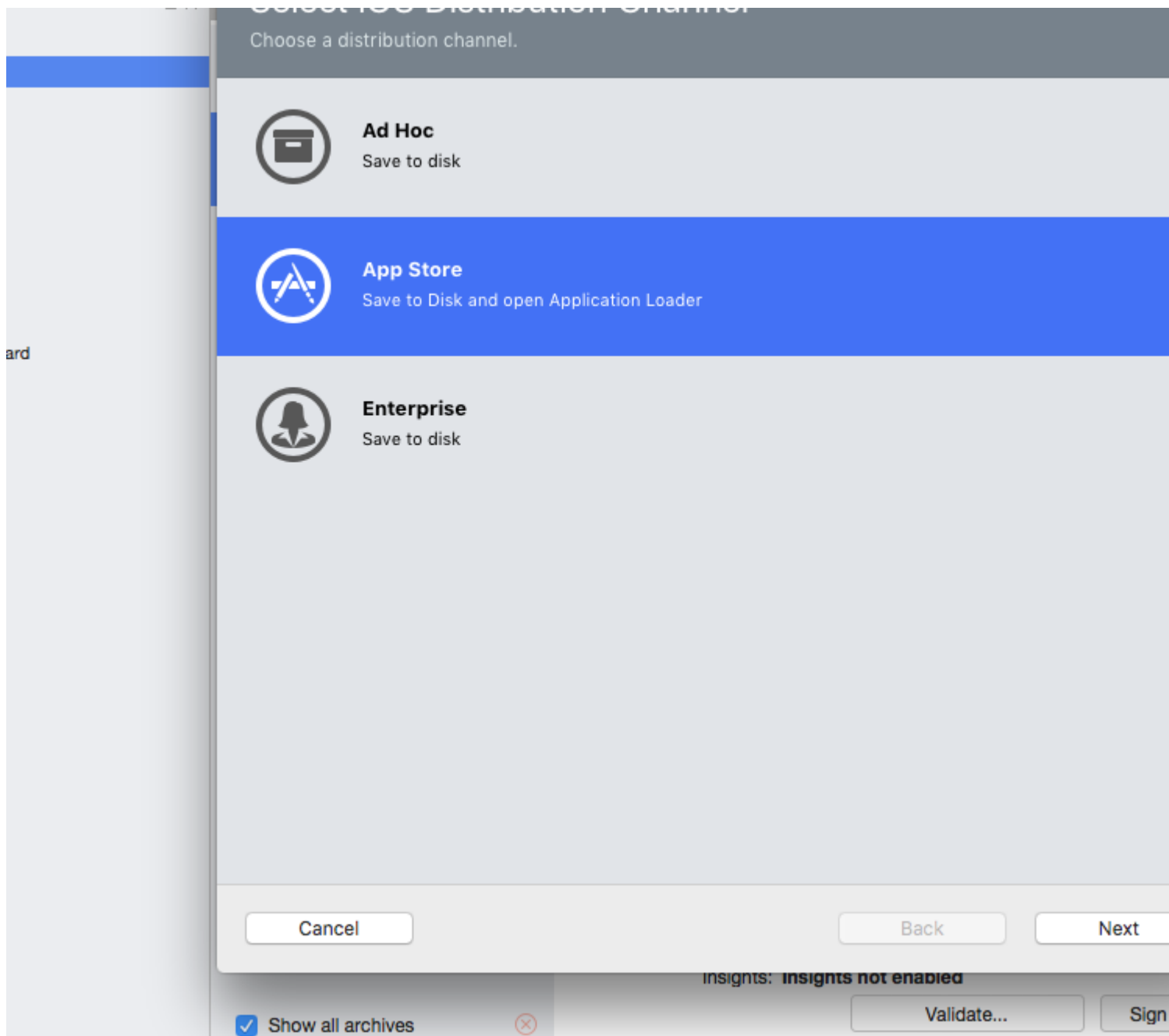
Apple exige que votre application soit envoyée avec un dossier SwiftSupport à côté de votre dossier Payload. Les deux sont dans votre paquet IPA.

Vous pouvez utiliser ce script <https://github.com/bq/ipa-packager> pour faire ce travail pour vous.

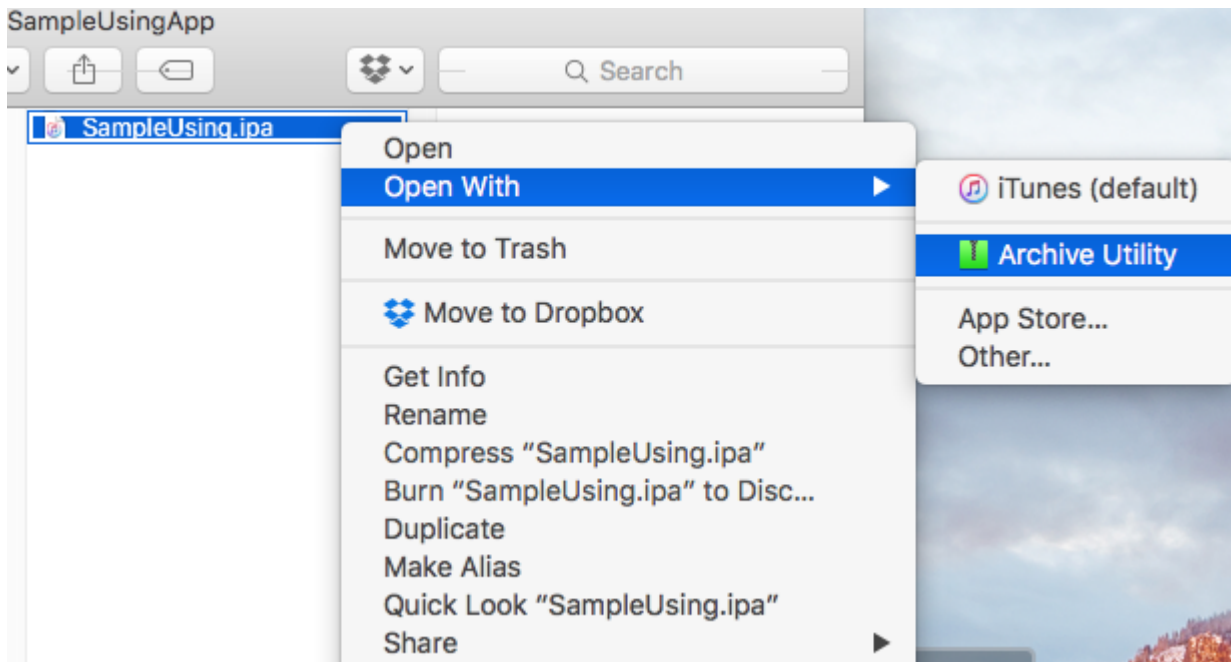
Ce processus est le seul que le consommateur de bibliothèque devra effectuer manuellement. Chaque fois qu'il / elle essaie de pousser l'application sur AppStore.



Cliquez sur "Sign and Distribute" et enregistrez sur le disque



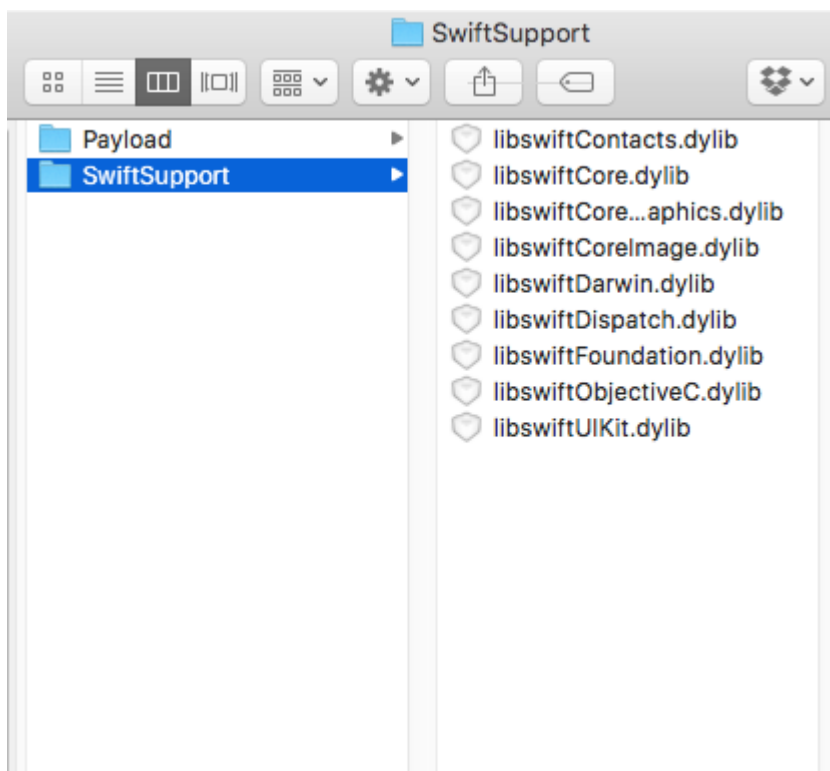
Décompressez votre .IPA



Créez le nouvel IPA en utilisant le script mentionné précédemment



Si vous décompressez le fichier, il contiendra le dossier SwiftSupport.



Remarques

Lors de la construction d'une bibliothèque dans Xcode, il est possible d'inclure les bibliothèques rapides. Ne pas! Ils seront inclus dans votre application finale sous `NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib`, mais ils doivent être inclus sous `NAME.app/Frameworks/libswift*.dylib`.

Vous pouvez trouver cette information ailleurs, mais cela vaut la peine de le mentionner: N'incluez pas le Bitcode dans la bibliothèque. À l'heure actuelle, Xamarin n'inclut pas Bitcode pour iOS et Apple exige que toutes les bibliothèques prennent en charge les mêmes architectures.

Avertissement

Ce guide est à l'origine créé par [Lucas Teixeira](#) . Tous les crédits lui appartiennent. Merci Lucas.

Lire Reliure des bibliothèques rapides en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6091/reliure-des-bibliotheques-rapides>

Chapitre 18: Tiroir de navigation Xamarin.iOS

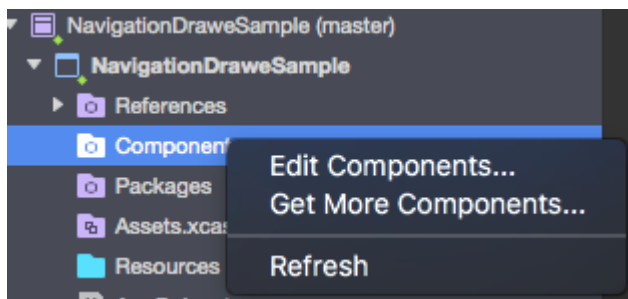
Syntaxe

1. Composant de navigation Flyout: <https://components.xamarin.com/view/flyoutnavigation>

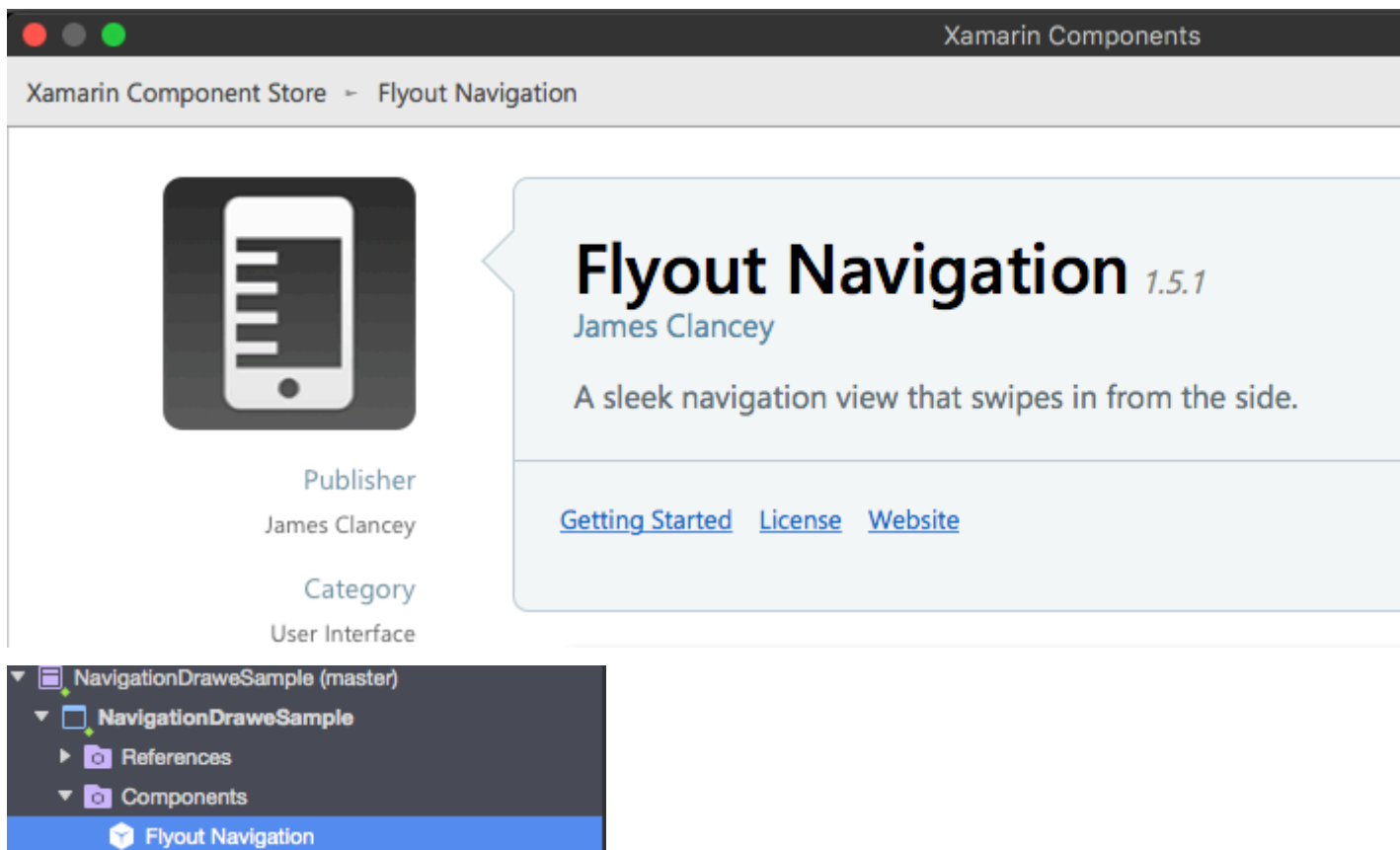
Exemples

Tiroir de navigation Xamarin.iOS

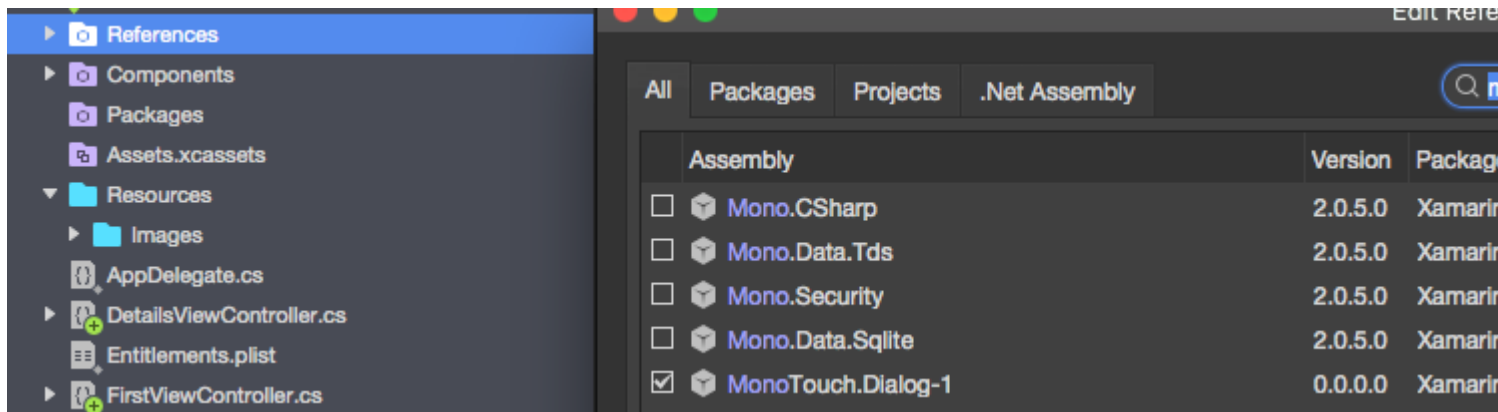
1. Créez un nouveau projet vide Xamarin.iOS (application Single View).
2. Faites un clic droit sur le dossier "Components" et sélectionnez "Get More Components":



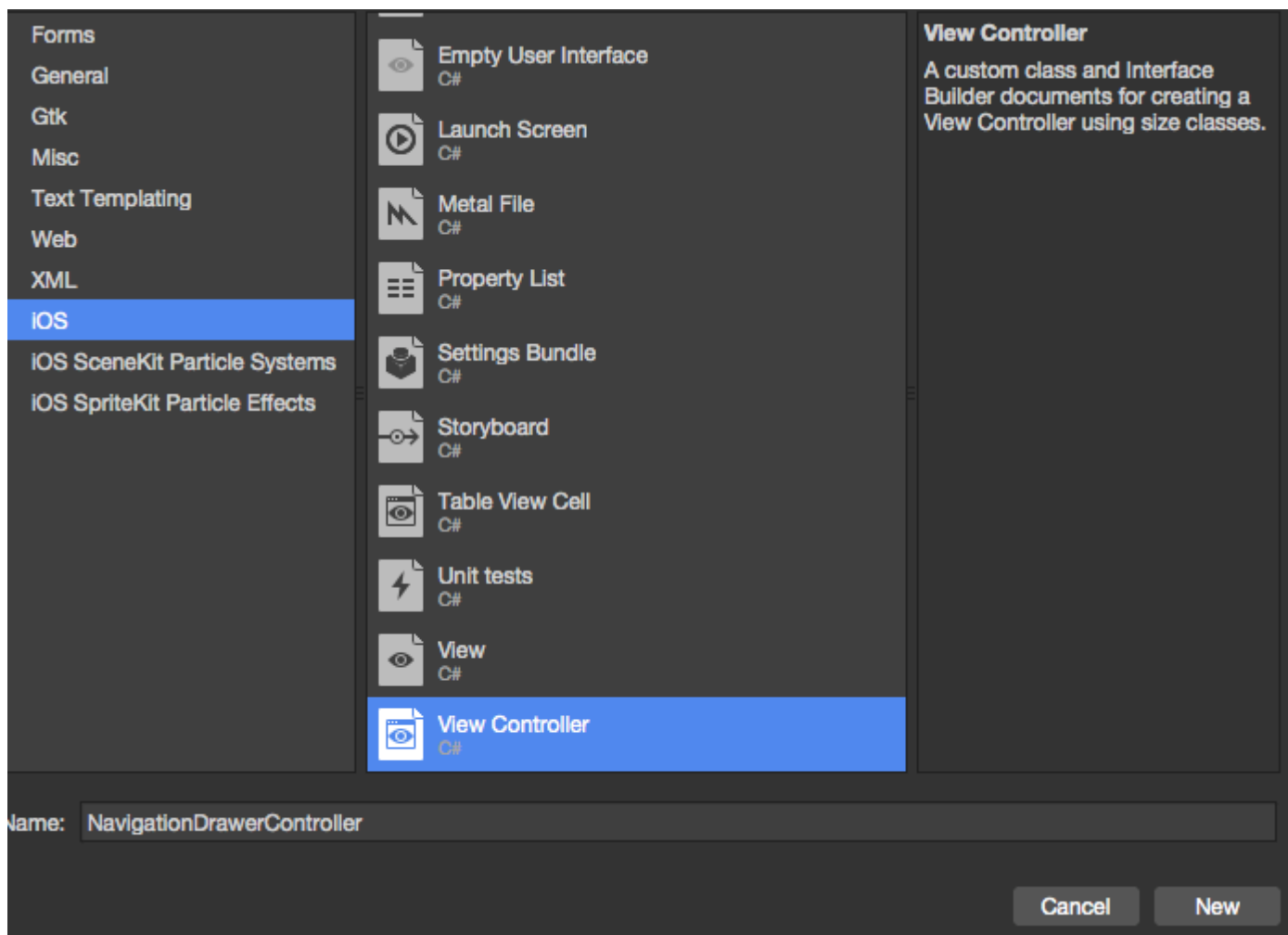
3. Dans la zone de recherche, tapez: "Flout Navigation" et ajoutez le composant ci-dessous à votre application:



N'oubliez pas également d'ajouter la référence "Mono.Touch.Dialog-1":



4. Maintenant, cliquez sur le projet et ajoutez le nouveau UIViewController appelé "NavigationDrawerController":



5. Maintenant, le code de la classe "NavigationDrawerController" devrait ressembler à ceci:

```
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;
}
}

```

Aucun souci que "MainNavigationController" soit surligné en rouge - nous l'ajouterons à l'étape suivante.

6. Ouvrez maintenant le fichier "Main.storyboard":

a) Ajoutez un UIViewController:

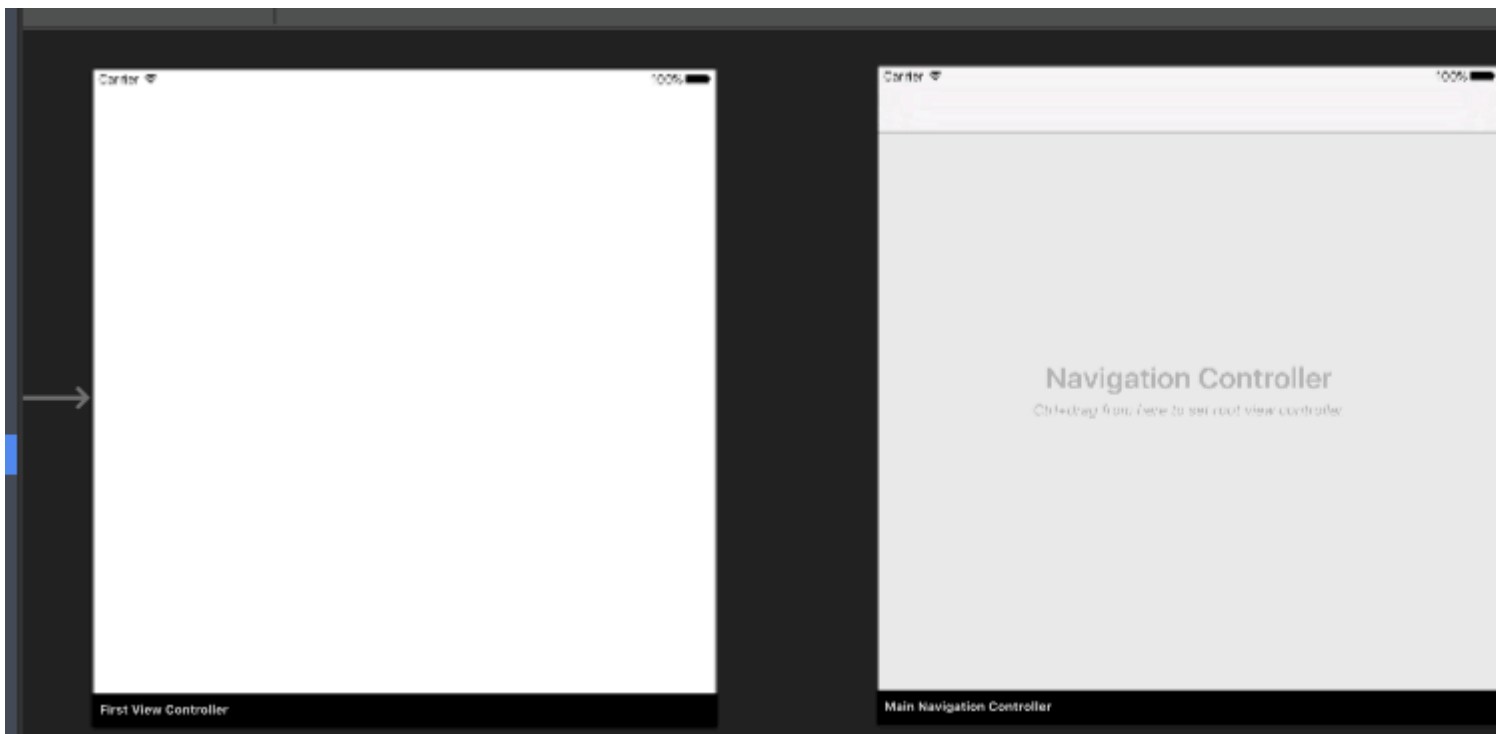
Remplissez les champs "Class" et "StoryboardID" avec ce nom: "FirstViewController"

b) Après cela, ajoutez le contrôleur de navigation avec la racine UIViewController:

Remplissez les champs "Class" et "StoryboardID" avec ce nom: "MainNavigationController" pour le contrôleur de navigation

Remplissez les champs "Class" et "StoryboardID" avec ce nom: "DetailsViewController" pour le contrôleur racine

Xamarin (ou Visual) Studio créera des classes code-behind pour les contrôleurs ci-dessus.



7. Maintenant, ouvrez la classe "FirstViewController" et collez ci-dessous le code:

```
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 defined for the drawer:
            NavigationRoot = new RootElement("Navigation")
            {
                new Section ("Pages")
                {
                    new StringElement ("MainPage")
                }
            },

            //Here are controllers defined for the drawer (in this case navigation controller
            with one root):
            ViewControllers = new[]
            {
                (MainNavigationController)Storyboard.InstantiateViewController("MainNavigationController")
            }
        };

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

```
}  
}
```

8. Ouvrez la classe "MainNavigationController" et collez ci-dessous le code:

```
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. La dernière classe appelée "DetailsViewController" devrait ressembler à ceci:

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

Veillez noter que "DetailsViewController" dérive de "NavigationDrawerController" que nous avons créé au début.

C'est tout. Maintenant, vous pouvez personnaliser le tiroir comme vous le souhaitez. Veuillez également trouver un échantillon prêt sur mon GitHub:

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

Lire Tiroir de navigation Xamarin.iOS en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6574/tiroir-de-navigation-xamarin-ios>

Chapitre 19: Travailler avec Xib et Storyboards dans Xamarin.iOS

Exemples

Ouverture de Xib / Storyboard dans Xcode Interface Builder à la place

Xamarin studio ouvre le fichier Xib et les storyboards par défaut dans Xamarin Designer. L'utilisateur peut faire un clic droit sur le fichier et `Open With -> `Xcode Interface Builder``

Lire [Travailler avec Xib et Storyboards dans Xamarin.iOS en ligne](https://riptutorial.com/fr/xamarin-ios/topic/6182/travailler-avec-xib-et-storyboards-dans-xamarin-ios):

<https://riptutorial.com/fr/xamarin-ios/topic/6182/travailler-avec-xib-et-storyboards-dans-xamarin-ios>

Chapitre 20: UIImageView zoom en combinaison avec UIScrollView

Remarques

Le UIImageView doit être dans une défilement pour que cela fonctionne.

La méthode DoubleTap bascule entre le minScale et le doubleTapScale.

Exemples

Tapez deux fois

```
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;
}
```

Geste de pincement 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; };
}
```

Lire UIImageView zoom en combinaison avec UIScrollView en ligne:

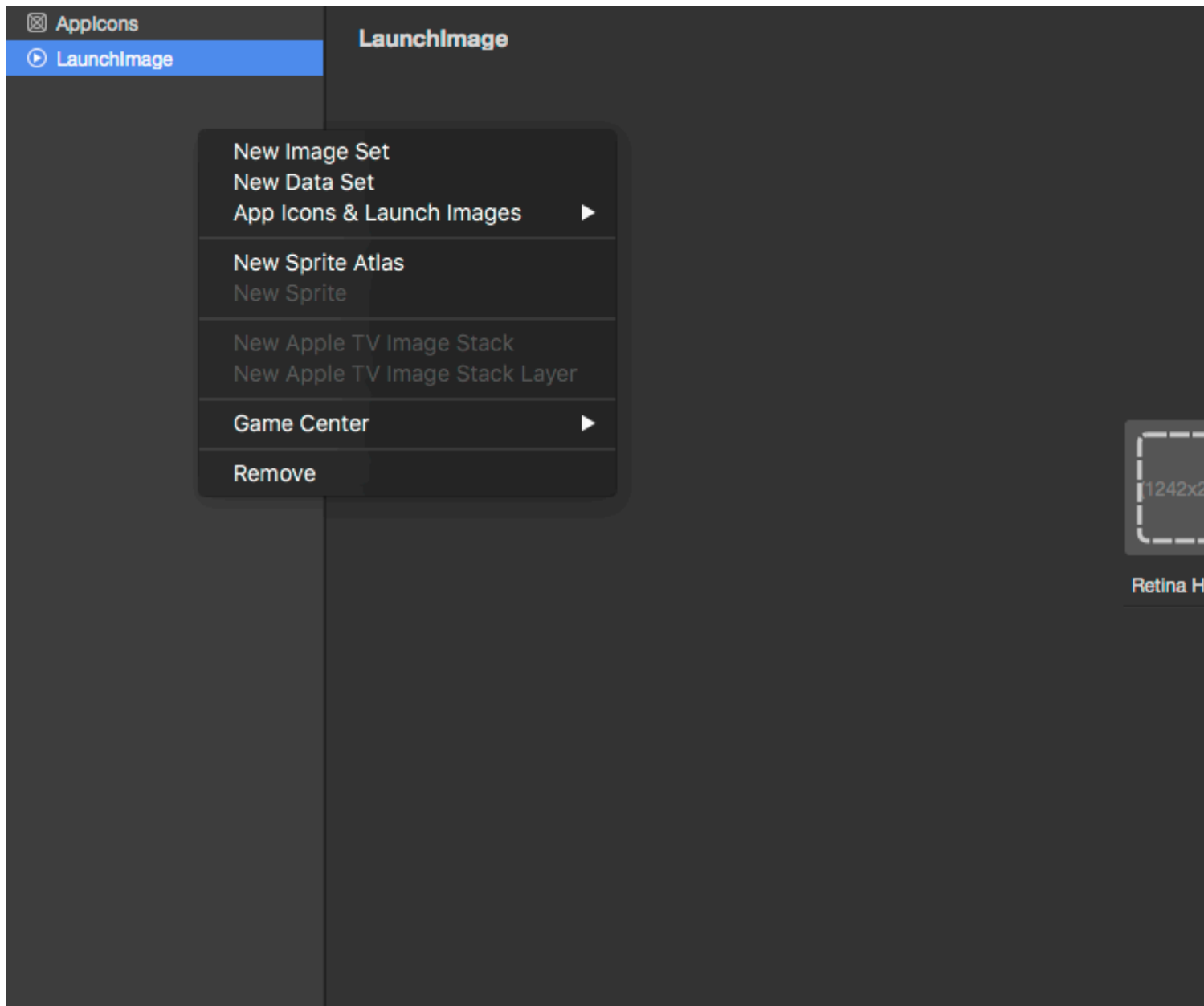
<https://riptutorial.com/fr/xamarin-ios/topic/6346/uiimageview-zoom-en-combinaison-avec-uiscrollview>

Chapitre 21: Utilisation de catalogues d'actifs

Exemples

Ajout d'éléments d'image au catalogue d'actifs

Voici à quoi ressemble le catalogue d'actifs dans Xamarin Studio,



Comme le montre l'image ci-dessus, il existe 5 types d'éléments que vous pouvez créer dans le catalogue.

Je ne couvrirai que les images, car c'est la plus simple.

Lorsque vous créez un nouvel ensemble d'images. Vous obtiendrez des options comme celle-ci

On-Demand Resource Tags:

Render As: Default



Vector

1x

2x

3x

Universal



Vector

1x

2x

R4

3x

iPhone



Vector

1x

2x

iPad



Vector

2x

38mm 2x

42mm 2x

Apple Watch



Vector

1x

2x

Mac

Pour ajouter des images au catalogue, vous pouvez simplement cliquer sur les carrés en pointillés

et sélectionner l'image que vous souhaitez définir pour une option particulière.

Dans XCode, vous avez les options 1x, 2x et 3x pour couvrir les tailles d'écran les plus récentes des appareils iOS. Mais Xamarin a une option supplémentaire Vector qui vous permet de télécharger une image vectorielle au format PDF qui serait automatiquement redimensionnée en fonction du périphérique sur lequel votre application s'exécute.

Pour les images iPhone, Xamarin conserve la taille d'image spéciale iOS 7 R4 utilisée pour les iPhone de taille d'écran de 4 pouces (5, 5S et SE).

Veillez vous référer à la [documentation de Xamarin pour savoir comment ajouter des images à une application iOS](#) pour plus d'informations.

Lire Utilisation de catalogues d'actifs en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/6630/utilisation-de-catalogues-d-actifs>

Chapitre 22: Utilisation de catalogues d'actifs iOS pour gérer des images

Remarques

Les catalogues d'actifs permettent de gérer plusieurs résolutions d'actifs d'image iOS. Pour afficher des images optimales, iOS utilise des versions 1x, 2x et 3x de chaque image en fonction de la densité d'écran de l'appareil. La version 1x ne concerne que les très anciens périphériques sans rétine, elle n'est donc pas nécessaire pour les applications ne supportant que iOS 9.

Les catalogues d'actifs vous aideront à gérer l'amincissement et le découpage des applications, optimisant ainsi les ressources que les utilisateurs doivent télécharger pour installer une application à partir de l'App Store.

Exemples

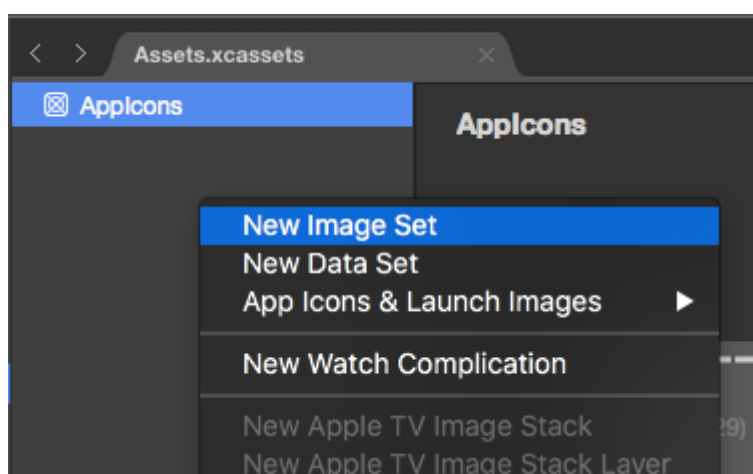
Chargement d'une image de catalogue d'actifs

Charger une image à partir d'un catalogue d'actifs à l'aide de `UIImage.FromBundle(string imageName)`

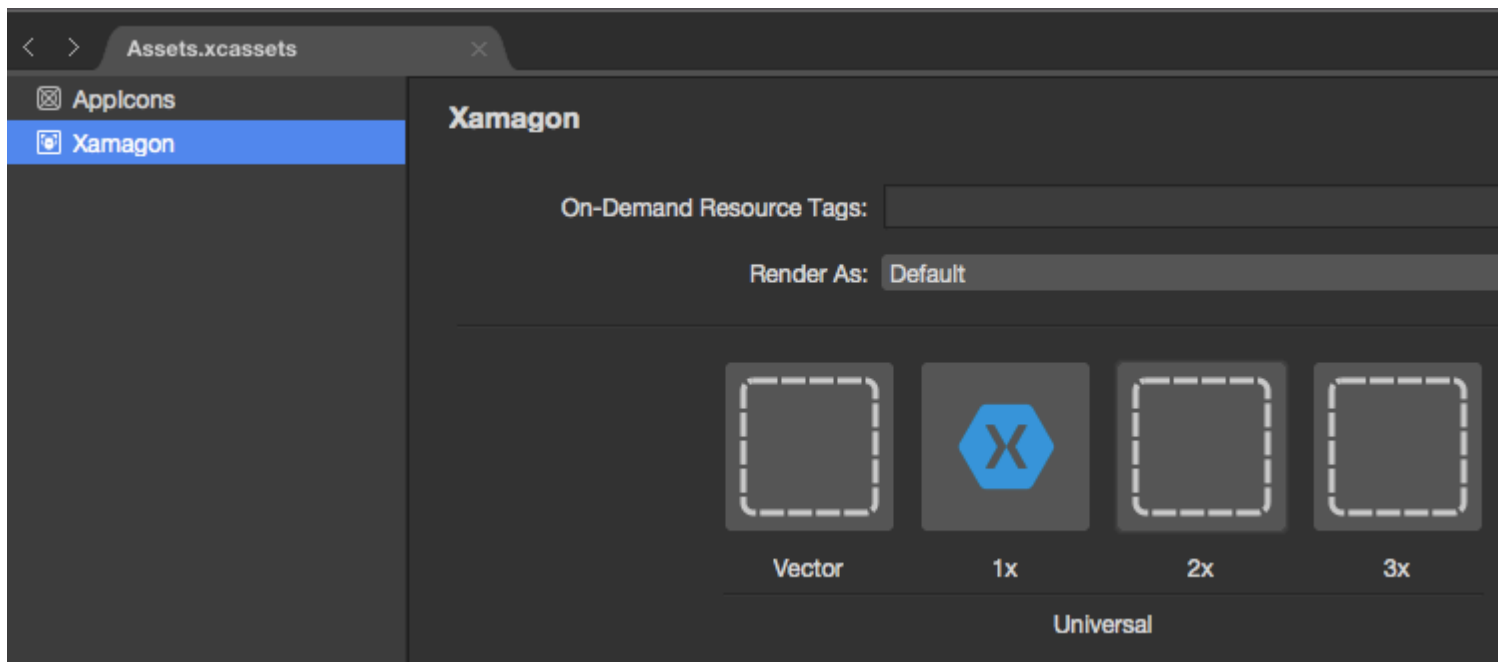
```
UIImage image = UIImage.FromBundle("ImageName");  
// use the name of the image set from the asset catalog
```

Vous pouvez utiliser l'image pour un `UIImageView` ou tout ce que vous devez faire.

Gestion des images dans un catalogue d'actifs



Les catalogues d'actifs permettent de gérer les images, les icônes d'application et les images de lancement. Image Set est utilisé pour les images affichées dans l'application. Les images universelles sont généralement la meilleure option. Vous pouvez soit utiliser une image vectorielle (telle que PDF) qui sera mise à l'échelle pour tous les écrans, soit inclure une variante 1x, 2x et 3x et iOS sélectionnera la version appropriée de l'image pour le périphérique actuel de l'utilisateur.



Vous pouvez modifier le nom de tout ensemble dans le catalogue d'actifs en double-cliquant sur le nom. Les images peuvent être ajoutées par glisser-déposer ou cliquer sur l'image que vous souhaitez insérer dans un sélecteur de fichiers.

Ajout d'images de catalogue d'actifs dans le storyboard

Les images de catalogues d'actifs peuvent être utilisées à partir de storyboards comme tout autre type d'image ajouté au projet. Ils seront automatiquement remplis en tant qu'option dans `UIImageView` et d'autres vues `UIImageView` en charge l'ajout d'une image.

Lire [Utilisation de catalogues d'actifs iOS pour gérer des images en ligne](https://riptutorial.com/fr/xamarin-ios/topic/6241/utilisation-de-catalogues-d-actifs-ios-pour-gerer-des-images):

<https://riptutorial.com/fr/xamarin-ios/topic/6241/utilisation-de-catalogues-d-actifs-ios-pour-gerer-des-images>

Chapitre 23: Xamarin iOS Google Insertion automatique

Introduction

Depuis que j'ai commencé à travailler avec Xamarin, il y a beaucoup de choses que j'aimerais que quelqu'un d'autre ait déjà documentées. Ici, j'explique comment utiliser 1 aspect de l'auto-complétion de Google Places, le contrôle de l'interface utilisateur utilisant un contrôleur de résultats. Bien que ce code soit basé sur des exemples de googles convertis en C # à partir de Swift, j'ai fait de mon mieux pour en faire un exemple pleinement fonctionnel. J'espère que cette documentation aidera beaucoup les autres.

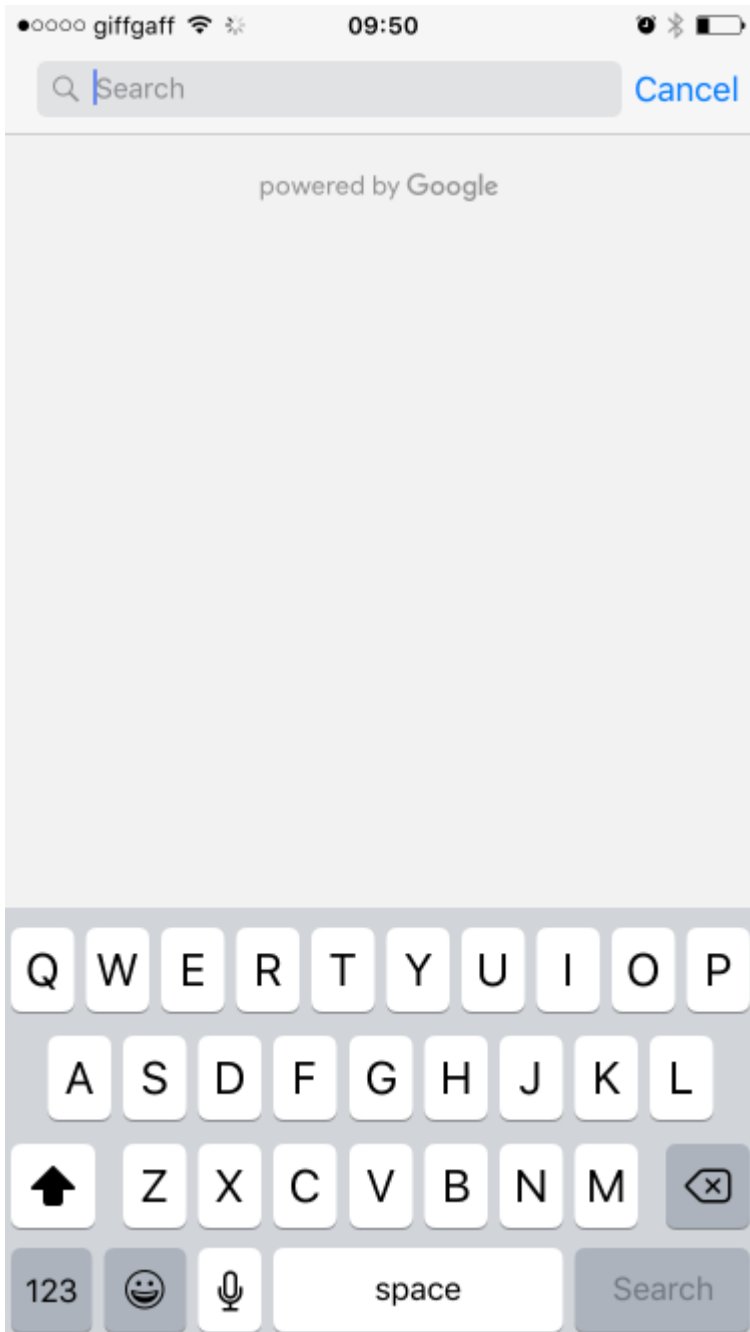
Exemples

Ajoutez un contrôle d'auto-complétion d'interface utilisateur avec le contrôleur de résultats.

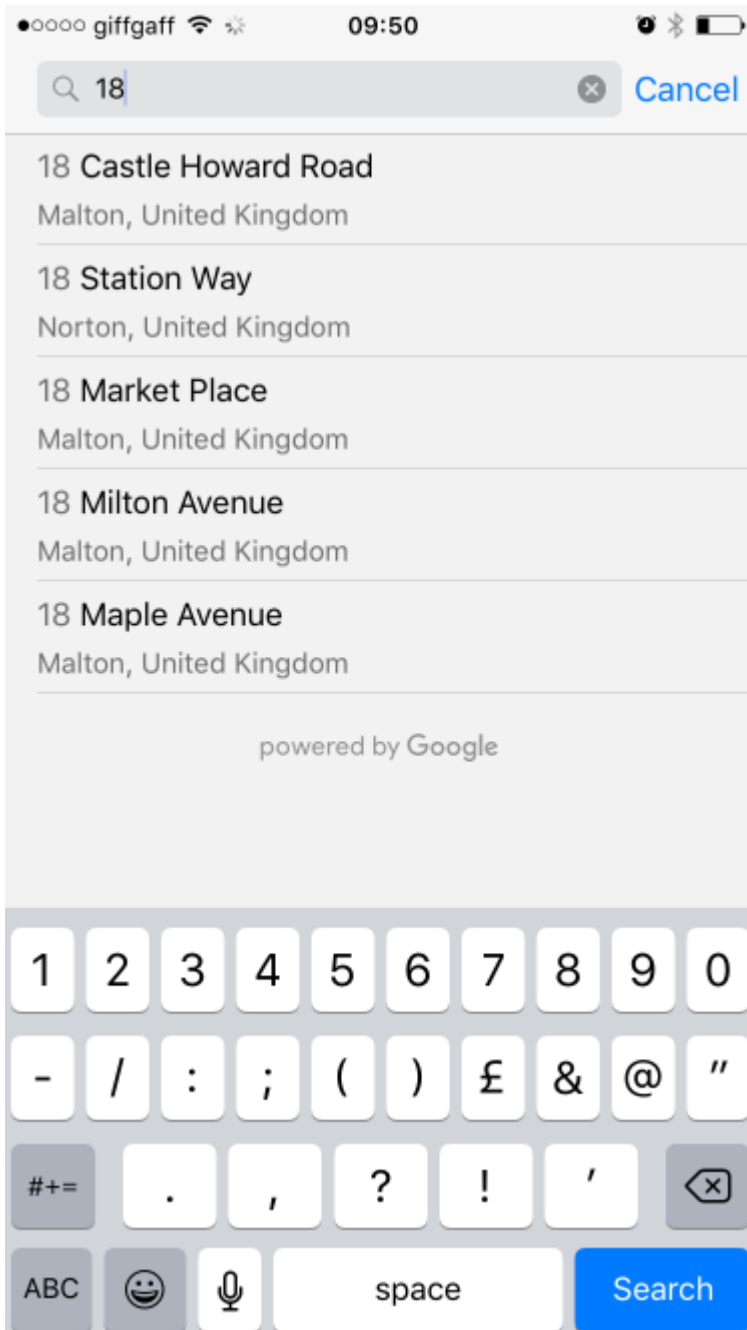
Le contrôle d'auto-complétion de l'interface utilisateur est une boîte de dialogue de recherche avec une fonctionnalité de saisie automatique intégrée. Lorsqu'un utilisateur entre des termes de recherche, le contrôle présente une liste de lieux à choisir. Lorsque l'utilisateur effectue une sélection, une instance `GMSPlace` (Place in Xamarin) est renvoyée, que votre application peut ensuite utiliser pour obtenir des détails sur l'endroit sélectionné.

Comme mentionné ci-dessus, cet exemple utilise un contrôleur de résultats qui permet de mieux contrôler l'interface utilisateur de saisie de texte. Le contrôleur de résultats bascule dynamiquement la visibilité de la liste de résultats en fonction du focus de l'interface utilisateur en entrée.

Le but de ce code est d'afficher un écran comme ci-dessous:



Cela va automatiquement compléter votre adresse lorsque vous commencez à taper, comme l'image ci-dessous:



Instructions:

1. Nous devons d'abord ajouter l'API Google Maps à notre Visual Studio. Elle est disponible via Nuget, recherchez simplement `Xamarin.Google.iOS.Maps`, ajoutez-la à votre projet iOS. Vous pouvez également la télécharger à partir de Xamarin [Xamarin Google Maps iOS SDK](#).
2. Nous avons besoin de quelque chose comme un bouton pour déclencher le contrôleur de vue autocomplete Google. Dans cet exemple, je le fais avec un story-board et j'ai ajouté un bouton nommé `GoogleButton`, vous pouvez le déclencher avec du code, cela n'a pas vraiment d'importance.
3. Sous `ViewDidLoad` dans votre classe de contrôleurs de vue, ajoutez le code suivant. Dans mon exemple ci-dessous, je n'utilise pas l'emplacement réel des appareils mobiles, ma solution finale le ferait bien sûr, mais c'était un test et je ne voulais pas implémenter de code supplémentaire avant d'avoir prouvé que cela fonctionnait ou diluer ce que j'essayais pour

vous montrer:

// Code pour faire apparaître le contrôleur de vue complète automatique de Google.

```
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. Ceci est facultatif, mais j'ai ajouté une fonction pour calculer les limites locales, je passe en 3000 ce nombre est en mètres, donc si vous voulez une plus grande limite initiale, n'hésitez pas à noter, s'il vous plaît dans le monde, il ne fait que pondérer les résultats initiaux à ces limites locales. Cette fonction a été empruntée à un message de dépassement de pile, je l'ai converti de Swift à C # pour nos besoins:

```
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. Cet extrait de code final est le délégué pour l'autocomplétion, nous avons besoin de ce délégué pour gérer tout ce que google va nous retourner:

```
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);
    }
}

```

Exécutez votre projet et il devrait fonctionner parfaitement, cet exemple est très ciblé, mais j'espère que cela vous donnera un exemple de base de la façon dont les contrôles de l'interface utilisateur de google autocomplete devraient fonctionner. Merci!

Lire Xamarin iOS Google Insertion automatique en ligne: <https://riptutorial.com/fr/xamarin-ios/topic/9041/xamarin-ios-google-insertion-automatique>

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec Xamarin.iOS	Amy Burns , chrisnr , Community , Dominic , hankide , Sergey , valdetero
2	Ajout de UIRefreshControl à une vue de table	manishKungwani , valdetero
3	Ajouter PullToRefresh à UITableView	Aditya Kumar , valdetero
4	Ajouter une barre de recherche à UITableView	Aditya Kumar , valdetero
5	Calcul de la hauteur des lignes variables dans GetHeightForRow	Larry OBrien , valdetero
6	Comment utiliser les catalogues d'actifs d'actifs	Aditya Kumar
7	Connexion avec Microsoft Cognitive Services	Daniel Krzyczkowski , valdetero
8	Contrôle de la capture d'écran dans le commutateur multitâche iOS	ben
9	Créer et utiliser des cellules de tableau prototype personnalisées dans xamarin.iOS à l'aide du storyboard	Daniel Krzyczkowski , valdetero
10	Des alertes	chrisnr , Gil Sand , patridge , Pilatus , Prashant C , valdetero

11	ID tactile	Amy Burns , ben , DannyC , Matthew , Peter Zhong , valdetero
12	Méthodes conseillées pour la migration d'UILocalNotification vers le cadre des notifications utilisateur	Aditya Kumar
13	Méthodes de redimensionnement pour UIImage	Frauke Nonnenmacher , raymondis , valdetero
14	Mise en page automatique dans Xamarin.iOS	ben , patridge , Tom Hawkin , valdetero
15	Programmation concurrente dans Xamarin.iOS	Ashan , Pilatus , Tom Gilder , valdetero
16	Reliure des bibliothèques rapides	Alex Sorokoletov , Elad Nava , Esam Sherif , J. Rahmati , James Mundy , Lucas Teixeira
17	Tiroir de navigation Xamarin.iOS	Daniel Krzyczkowski , valdetero
18	Travailler avec Xib et Storyboards dans Xamarin.iOS	lukya
19	UIImageView zoom en combinaison avec UIScrollView	Citroenfris , valdetero
20	Utilisation de catalogues d'actifs	aniket.ghode , mnoronha
21	Utilisation de catalogues d'actifs iOS pour gérer des images	dylansturg , valdetero
22	Xamarin iOS Google Insertion automatique	Conrad