

# プログラミング入門 iOS 11+Xcode 9対応





# 本PDFについて

- 本 PDFは書籍『Swift 4 プログラミング入門 iOS 11 + Xcode 9 対応』(ISBN978-4-8026-1153-4)
   をご購入いただいたお客様のために「ボーナス PDF」として作成されています。
- ●本 PDFの最新版は上記書籍の<u>サポートサイト</u>より入手できます。本 PDFの内容は予告なく更新されることがあります。最新の情報につきましては、サポートサイトをご覧ください。
- ●本 PDFは、内容と形態を変更しない限り、自由に配布してくださって構いませんが、上記書籍のご購入を前提とした説明が多く含まれます。予めご了承ください。
- ●本PDF中の表現で「書籍版」とあるのは、上記書籍を指します。
- ●本PDFの内容は無保証です。本PDFを販売することを禁じます。
- 本 PDF を印刷することはできません。
- PDF 中の URL はハイパーリンクとして機能します。
- PDF 中の各項目には「しおり」が設定されています。目次の代わりにお使いください。

# サンプルプログラムについて

サンプルプログラムは、以下からダウンロードできます。

https://github.com/tnantoka/swiftbook-examples/archive/1.0.zip

MITライセンスです。自己責任になりますが自由にご利用くださって構いません。また、配布している サンプルコード内にある「/// [marker数字 ]」というコメントは執筆上のマーキングであり、プログラ ム上の意味はありません。

一部のサンプルでは、以下のフォントを使用しています。

- <u>http://fontawesome.io/</u> (SIL Open Font License)
- https://fonts.google.com/specimen/Italiana (SIL Open Font License)

本PDFでサンプルプログラムを示す場合は、以下のようにサンプルプログラムの場所を示します。

# 

種類	ファイル
プロジェクト	Bonus/Chapter1/Icon.xcodeproj
Storyboard	Bonus/Chapter1/Icon/Icon/Base.Iproj/LaunchScreen.storyboard
ソース	変更しません

# 『Swift 4プログラミング入門』ボーナスPDF

# App Storeにリリースする

Chapter

本 PDFの Chpater 1 では作成したアプリを App Store にリリースするための手順を紹介します。

Section 1.1	<u>アイコン・起動画面の準備</u>
Section 1.2	デザインを整える
Section 1.3	サポートサイトの開設
Section 1.4	
Section 1.5	 レビューガイドラインの注意点





# 

種類	ファイル
プロジェクト	Bonus/Chapter1/lcon.xcodeproj
Storyboard	Bonus/Chapter1/Icon/Icon/Base.Iproj/LaunchScreen.storyboard
ソース	変更しません

アプリをインストールすると、デフォルトでは以下のようなアイコンがホーム画面に表示されます。また、 起動画面については真っ白です。独自のものを設定しましょう



iPhoneアプリには次の表にあるサイズのアイコンを設定する必要があります。

サイズ	必須
40px	_
58px	_
60px	_
80px	-
87px	—
120px	0
180px	0
1024px	○ (Xcode9から)

必須に○がついているものは、設定しないと Appleへ提出する際エラーになります。他のものもできるだけ設定しておいた方がいいでしょう。

今回は以下のようなアイコンを必須の3サイズ(120px、180px、1024px)用意しました。

$\mathbf{C}$	
<b>)</b>	

#### Note

残念ながらXcodeにアイコンの作成機能は備わっていないので、画像編集ツールで作成しましょう。 筆者は、書籍版の Chapter 15で紹介した図形描画を応用し、本 PDFの Chapter 3で紹介する Xcode Playgroundでアイコンを作成しました。

https://github.com/tnantoka/lconCreator

アイコンの設定はAssets.xcassetsのAppIconから行います。



それぞれのサイズのところにドラッグ&ドロップして設定します。サイズが少しわかりづらいですが、それ ぞれ以下のサイズのアイコンを指定します。

- $\lceil 60pt \rfloor \mathcal{O} \lceil 2x \rfloor : \lceil 60 \times 2 = 120px \rfloor$
- $\lceil 60pt \rfloor \mathcal{O} \lceil 3x \rfloor : \lceil 60 \times 3 = 180px \rfloor$
- [1024pt」の [1x]: [1024×1=1024px]



この状態で実行すると、ホーム画面のアイコンが独自のものに変わります。



次は起動画面を変更します。デフォルトでは以下のように真っ白になっています。これを中央にアイコンを 置いたシンプルなものに変更します。





「Assets.xcassets」に新規画像セットを追加します。名前は「Launchlcon」に変更します。





「2x」「3x」に、それぞれ「120px」「180px」のアイコン画像を設定します。「iPhone アプリに必要なア イコン」で使った画像と同じです。



設定は「LaunchScreen.storyboard」で行います。操作方法は普通のStoryboardと一緒です。 Image Viewを中央に配置します。どのサイズのデバイスでも中央に表示されるように AutoLayoutで制 約を設定します。



Image Viewに画像を設定します。Utility Areaの「Image」に先ほど追加した「Launchlcon」を設定します。



Image Viewの背面にあるViewの背景色をアイコンの背景色と揃えます。アイコンの背景色はRGB(237, 237, 237)で制作してあります。



これでアプリの起動時に独自のLaunchScreenが表示されるようになります。



# ゙ サンプルアプリ …………



# 

種類	ファイル
プロジェクト	Bonus/Chapter1/Appearance.xcodeproj
Storyboard	Bonus/Chapter1/Appearance/Appearance/Base.lproj/Main.storyboard
ソース	Bonus/Chapter1/Appearance/Appearance/AppDelegate.swift
	Bonus/Chapter1/Appearance/Appearance/FirstViewController.swift
	Bonus/Chapter1/Appearance/Appearance/NavigationController.swift

# デザイン変更の基礎知識 ………

開発メンバーにデザイナーがいる場合は、ぜひデザインを依頼し、そのデザインに合わせて開発を進めましょう。

デザイナー以外が下手に手を加えると、何も手を加えない状態よりも使いにくくなってしまう可能性もあり ますから、注意しましょう。開発メンバーにデザイナーがいない場合は、色をアイコンに合わせる程度の調整 で留めておくのもよいでしょう。

ここでは各種バーの色を変える手順を紹介します。

#### Note

デザインを変更する際は、Apple社が定めたガイドラインから外れないように気をつけてください。 『Human Interface Guidelines』 https://developer.apple.com/ios/human-interface-guidelines/overview/themes/

UIAppearance .....

UIAppearanceは各UI部品のデフォルト設定を変えるときに使うクラスです。

各UI部品が表示される前に設定を変更する必要があるので、AppDelegateのapplication(\_:didFin ishLaunchingWithOptions:)内に設定を変更する処理を書きます。次のコメントが目印です。

// Override point for customization after application launch.

UIApperanceの他にも、データベースの初期化などアプリ起動時に行いたい処理はこの場所に記述することが多いです。

UIAppearanceの書式は次の通りです。

書 式 対応バージョン (3) (4)

クラス名.appearance().変更するプロパティ = 設定値

以降ではUIAppearanceの機能を使ってバーの色を変更します。



3つのバーがあるアプリを作成します。

- タブバー(Tab Bar)
- ナビゲーションバー(Navigation Bar)
- ツールバー (Toolbar)

設定を簡単にするため、ここでは「Tabbed App」テンプレートを使います。



これだけで、タブバーを表示させることができました。

次に、ナビゲーションバーを表示させるために、「First」ビューコントローラーを「Navigation Controller」 へ埋め込みます。「Navigator Area」の「Project Navigator」で「Main.storyboard」を選択し、「Document Outline」で「First View Controller Scene」の左側の三角形をクリックします。表示された「First」ビュー コントローラーを選択し、メニューバーから「Editor」→「Embed In」→「Navigation Controller」を選 択します。



Editor Product Debug Source Co	ontrol Window Help	1
Canvas Zoom	at 21:54	
Hide Document Outline Reveal in Document Outline	oard > Main.storyboard (Base) > First Scene > First > View	
Align     ►       Arrange     ►       Size to Fit Content     %=       Update Frames     ℃%=		
✓ Snap to Guides Guides		
Embed In  Unembed Localization Locking	View Scroll View Stack View	$x = -1$ $\rightarrow$ $b$ [Editor] $\rightarrow$ [Embed in]
✓ Automatically Refresh Views Refresh All Views Debug Selected Views	Navigation Controller Tab Bar Controller	→ [Navigation Controller]の順で選択します。
Resolve Auto Layout Issues		
Refactor to Storyboard		
	First View Loaded by FirstViewController	

さらに、ツールバー(Toolbar)を表示させます。「Main.storyboard」で「Navigation Controller」 を選択した状態で、「Utility Area」の「Shows Toolbar」をオンにします。



バーだけですと味気ないので、ナビゲーションバーとツールバーにアイテム(Bar Button Item)を追加し ます。ここでは見た目のカスタマイズをしたいだけですので、特にアクションなどは設定しません。 ツールバーに「ゴミ箱(Trash)」ボタンを追加してみます。





これで、ツールバーに「ゴミ箱」ボタンを追加できました。

続いてナビゲーションバーに、「作成(Compose)」ボタンを追加してみます。



Chapter 1	App Store にリリースする

Bar Button Item	I	
Style	Bordered 🗘	 
System Item	Compose 🗘	● 側に□が表示されている状態)で、「Utility Area」
+ Tint	Default 🗘	の「System Item」を「Compose」に変更します。
Drag and Drop	Spring Loaded	

これで、ナビゲーションバーに「作成」ボタンを追加できました。

最後に、ナビゲーションアイテム(Navigation Item)にタイトルをつけておきます。



以上で準備は終わりです。何も設定しないと3つのバーの背景色はすべて薄いグレーで表示されています。



ナビゲーションバーのデザイン変更から始めます。「Navigator Area」の「Project Navigator」で「AppDelegate.swift」を選択し、Section 21.2の先頭で目印としたコメントを探し、次のプログラムを入力します。

サンプルプログラム Bonus/Chapter1/Appearance/Appearance/AppDelegate.swift

```
// Override point for customization after application launch.
UINavigationBar.appearance().barTintColor = .darkGray
UINavigationBar.appearance().tintColor = .white
UINavigationBar.appearance().titleTextAttributes = [
    NSAttributedStringKey.foregroundColor: UIColor.white
]
```

barTintColorが背景色、tintColorが左右のボタンの色、titleTextAttributesは中央のタイト ルの設定です。

このように設定すると、ナビゲーションバーの色が変わります。



## 注意

UINavigationBar全体のデザインを変えると、予期せぬ部分に影響を与えてしまう可能性があります。 例えば、UIImagePickerControllerのバーも色が変わってしまいます。



Carrier 🗢	11:16 PM	-	
	Photos	Cancel	意図
T	Moments	>	
	Camera Roll 5	>	
My Album	ıs		
[			

意図しない場所の色も変わってしまいました。

UINavigationControllerを継承した NavigationController を使えば、自分の作成したバーだけ にAppearanceを設定することができます。

メニューバーから「File」→「New」→「File」と選択し、新規ファイルのテンプレート選択画面を表示 させます。表示された画面で「Cocoa Touch Class」テンプレートを選択し、「Next」ボタンを押します。



保存先を指定するダイアログが表示されるので、プロジェクト名と同じ名前のフォルダーを指定して 「Create」ボタンをクリックすれば、NavigationController.swiftを作成できます。

「Navigator Area」の「Project Navigator」で「Main.storyboard」を選択し、「Navigation Controller」 を選択します。

「Utility Area」上部に並ぶボタンのうち左から3番目の「Show the identity inspector」ボタンをクリックし、カスタムクラスを指定します。

Section 1.2 デザインを整える

カスタムクラス配下のナビゲーションバーのみにデザインの変更が反映されるように Appearanceの書き 方を変更します。書式は以下のとおりです。

書 式 対応バージョン (13) (14)

クラス名.appearance(whenContainedInInstancesOf: [対象のクラス. self]).変更するプロパティ = 設定値

サンプルプログラム Bonus/Chapter1/Appearance/Appearance/AppDelegate.swift

UINavigationBar.appearance(whenContainedInInstancesOf: [NavigationController.self]).barTintColor = .darkGray UINavigationBar.appearance(whenContainedInInstancesOf: [NavigationController.self]).tintColor = .white UINavigationBar.appearance(whenContainedInInstancesOf: [NavigationController.self]).titleTextAttributes = [ NSAttributedStringKey.foregroundColor: UIColor.white ]

これで、UIImagePickerControllerはデフォルト色のままになりました。



# ToolBarのデザイン変更 ………

ツールバーもナビゲーションバーと同様にデザインを変更できます。

サンプルプログラム Bonus/Chapter1/Appearance/Appearance/AppDelegate.swift

UIToolbar.appearance().barTintColor = .darkGray
UIToolbar.appearance().tintColor = .white



TabBarのデザイン変更 ………

タブバーもナビゲーションバーと同様にデザインを変更できます。

サンプルプログラム Bonus/Chapter1/Appearance/Appearance/AppDelegate.swift

UITabBar.appearance().barTintColor = .darkGray
UITabBar.appearance().tintColor = .white



このようにバーの色を変更するだけでもアプリのイメージが変わります。

#### Note

バーの色を変えたあとはボタンの色との相性を必ず確認しましょう。バーの色だけを変えるとボタンが見づ らくなってしまう場合があります。



アプリを申請するためには、ユーザーが連絡の取れるサポートサイトが必要です。

企業であればコーポレートサイトを指定しておけば問題ありませんが、アプリ専用のランディングページな どを用意する方が望ましいでしょう。Twitterアカウントや Facebook ベージを指定しても大丈夫なので、 手間をかけたくない場合はそれでも構いません。

# GiHub Pages .....

GitHub Pagesを使えば無料でサポートサイトをホスティングしてもらえます。日常的にホスティング サービスなどを利用していない場合には検討してみてください。



Section 1.3 サポートサイトの開設

Netlify .....

筆者は無料で独自ドメインのHTTPS化もできる Netlify を利用しています。

● ● < > □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	e netify.com	C.	Sign Up	[Netlify] https://www.netlify.com/
Write frontend code All the features developers Deployme	e. Push it. W need right out of the box: Gi nt, one click HTTPS and mu Get started for free Partie free view or oracle and reasons.	e handle th Iobal CDN, Continuous ore	e rest.	
Excelling a generalized with a generalized with the generalized wit	alamata dadgarar gashady databatag timen parasa wana.	•	0	

WordPress.comなどのブログや CMS(コンテンツマネージメントシステム)のホスティングサービス を使う選択肢もあります。

このタイプのサービスは広告が表示される代わりに無料で利用できるというものが多いです。





# 

アプリには App ID というものを割り当てる必要があります。App IDは Apple Developerのサイトにロ グインして作成します。

その際にが、アプリを一意に識別する Bundle ID という情報が使われます。

Bundle IDはプロジェクト作成時に入力した情報から自動的に作成されますが、後から変更することもできます。

Choose options for your new project:			
Product Name:	AppStore	)	このようにプロジェクトを作成した提合
Team:	None		
Organization Name:	tnantoka	]	com.tnantoka.AppStore」か Bundle ID
Organization Identifier:	com.tnantoka		となります。
Bundle Identifier:	com.tnantoka.AppStore		
Language:	Swift		
	Use Core Data		
	Include Unit Tests		
	Include of fests		
Cancel	Pre	vious Next	

プロジェクト作成後に Bundle IDを変更するには、「Navigator Area」の「Project Navigator」で、 一番上の「プロジェクト」を選択します。

(1	D <b>[TARGE</b> ⁻	TS」	『プロジェクト名と同	同じターク	ゲットを選	択します。		
	PROJECT AppStore TARGETS AppStore	Genera	Capabilities Resource Ta	gs Info AppStore com.tnantoka.A 1.0 1	Build Settings	Build Phases	Build Rules	— ② [General]を選択します。
	③ [Bu	Indle	Identifier」を書き換	えること	でBundle I	D を 変更す	ることができます。	

設定したBundle IDがわからなくなった場合には、この画面を確認するとよいでしょう。

# 審査提出の基本的な流れ ………………………………………………………………

審査提出は次のような流れで進めます。Apple社のアプリ申請Webサイトのデザインは変更されることがありますが、申請提出の流れはほとんど変わっていません。

- 1. App ID 作成
- <mark>2</mark>.チーム設定
- 3. バージョン作成
- 4. Archive して送信
- 5. スクリーンショット
- <mark>6</mark>. 提出

#### Note

申請にあたっては、最新バージョンの Safariを使うのが無難です。Chromeや Firefox では正しく動かない ことがあるためです。

# 

Apple開発者用のサイト(<u>https://developer.apple.com/account/</u>)でApp IDを作成します。

Apple Inc.	â Ø +	
C Developer		
		①Apple IDとパスワードを入力します。
Eurgot ID of Password?		

#### Chapter 1 App Store にリリースする









	Apple Inc.		<u>د</u>	ð	ĺ
= All	HealthKit:	© Disabled			
Apple TV	HomeKit:	© Disabled			
Apple Watch	Hotspot:	© Disabled			
= iPad	iCloud:	© Disabled			
iPad Touch	In-App Purchase:	Enabled			
	Inter-App Audio:	© Disabled			
Provisioning Profiles	Multipath:	© Disabled			
= Development	Network Extensions:	© Disabled			
Distribution	NFC Tag Reading:	© Disabled			
	Personal VPN:	© Disabled			
	Push Notifications:	© Disabled			
	SiriKit	© Disabled			
	Wallet:	© Disabled			
	Wireless Accessory Configuration:	© Disabled			
		_	_		
		Don			-
	Copyright ID 2017 Apple Inc. All rights reserved. Terms	of Use Privacy Policy			

# 

App Store で配布するアプリは、証明書を使って署名をする必要があります。難しそうに聞こえますが Xcode がほぼ自動でやってくれます。その準備として書籍版の Chapter 19の実機検証と同様に Teamを 選択しておきます。



アプリをアップロードするためには、iTunes Connect上でアプリのバージョンを用意しておく必要があ ります。

iTunes Connect (<u>https://itunesconnect.apple.com/login</u>) にログインします。

	₿ Apple Inc.	¢	<u>a</u> = +	
iTunes Connect			0	
	iTunes Connect	<b>0</b>		①Apple IDとパスワードを入力します。 ②[→]をクリックします。
き Control Control C	Cox 単 Action Inc. こ」、スイスで利用り組になりました。	yright © 2017 Apple Inc. All s	ights reserved   Privacy Policy	
スタリカ合類国、イギワス、マ・ ロモンランブであるのキッシ トを提用して Saurch Ads を料 して、 マイ App		メキシコ、スイスの App /パーンを設定する際に、	Store そ App を 7 100 UEDのクレジッ	
▲	を約了作品が この的な 2007 KENE K APPEN	A & A	j Anwey Novey   Contrict Un	──  マイ App]を選択します。



必要事項を入力します。

「名前」は AppStore での表示名で、すでに登録されるアプリの名前と重複した名前を付けることはできません。ここでは「AppStore」がすでに使われていたため「AppStoreサンプル」という名前で登録することにしました。

「バンドルID」では、先ほど作成したApp IDを選択します。



「1.0 提出準備中」と表示されれば、バージョンの作成は完了です。

••• <>	Apple Inc.	Ċ	1 0 +	
iTunes Connect マイ App ~	AppStoreサンプル ~	飛) Tatsuya	岡辰哉~   ⑦ Tobioka	
App Store 機能 TestFlight	オー アクティビティ			
APP STORE 情報 App 情報 価格および配信状況	App 情報 この情報は、App のすべてのブラットフォームで使 ージョンでリリースされます。	用されます。 変更内容は次の App	保存	
iOS APP © 1.0 提出準備中	ローカライズ可能な情報	the state which was a	日本語~②	
(+) パージョンまたはブラットフォーム	AppStoreサンプル	http://example.com (オプション		
0	18 <b>サブタイトル ①</b> オブション 30			
	ー 般情報 パンドルID 参 新しいパンドル ID を登録する AppStore - com.tnantoka.AppStore ↓ ↓	プライマリ言語 ⑦ 日本語		
	あなたのバンドル ID com.tnantoka.AppStore	カテゴリ ②		

Note

バージョンアップの場合は左下の「+バージョンまたはプラットフォーム」からバージョンを追加します。

# 

リリースするアプリはアーカイブ機能によって ipaというファイルにまとめる必要があります。そのファ イルをiTunes Connectにアップロードします。

# Archive

Xcodeでアプリをアーカイブします。

Apps No devices connect Build Only Device Ceneric IOS Device IOS Simulators Pad (5th generatic	ted to 'My Mac'
Build Only Device     Build Only Device     Ceneric IOS Device     IOS Simulators     iPad (5th generatic     iPad Air	
Generic IOS Device	
iOS Simulators iPad (5th generation) iPad Air	
iPad (5th generatio	
iPad Air	n)
iPad Air 2	
iPad Pro (9.7-inch)	
rd iPad Pro (10.5-inc)	1)
iPad Pro (12.9-inch	) (2nd generation
iPhone 5s	,
iPhone 6	
🏢 iPhone 6 Plus	
iPhone 6s	
iPhone 6s Plus	
iPhone 7 Plus	
V iii iPhone 8	
iPhone 8 Plus	
iPhone SE	
iPhone X	
Add Additional Simulator	s
Download Simulators	
	Source
Debug	Source
Run	ЖR
Test	¥11
Drafile	000
Profile	жι
Analyze	<b>企業B</b>
7 (1 K) Y Z O	
Archive	
Archive	
Archive Build For	
Archive Build For	
Archive Build For Perform Action	•
Archive Build For Perform Action	•
Archive Build For Perform Action Build	► ₩B
Archive Build For Perform Action Build Clean	► 米B 企業K
Archive Build For Perform Action Build Clean Stop	► #B ☆#K
Archive Build For Perform Action Build Clean Stop	►
Archive Build For Perform Action Build Clean Stop	₩B ☆₩K ₩.
Archive Build For Perform Action Build Clean Stop Scheme Destination	₩B ☆₩K ₩.
Archive Build For Perform Action Build Clean Stop Scheme Destination	►
Archive Build For Perform Action Build Clean Stop Scheme Destination	► #B ☆ #K #.
,	

アーカイブが終わると Organizer というウィンドウが開きます。アップロード作業はこのウィンドウで行います。

#### Note

Organizerのウィンドウを閉じてしまった場合には、メニューバーから「Window」→「Organizer」を選 択することで起動することができます。



提出用の署名に使う証明書を作成する必要があります。この作業が必要なのは初回アップロード時のみです。

#### Note

書籍版のChapter 19の実機テストのときに作ったのは開発用の証明書で、こちらは本番用の証明書です。



証明書を Export しておくことを勧められます。Export したファイルを他の Mac で開けば、その Mac でもリリース作業を行うことができます。Export してみます。

Previous Next

選択します。

② [Next]をクリックします。

Generate an iOS Distribution certificate

Cancel

is required to distribute your application.

Export signing certificate:	
Xcode has created a new IOS Distribution signing certificate for "Tatsuya Tobioka". The private key for this certificate is stored locally in your Keychain and cannot be recovered if lost. Apple recommends exporting the certificate and private key and storing them in a safe place.	
Export Signing Certificate	「Export Signing Certificate」をクリックします
1	
Cancel Previous Next	

Chapter 1	App Store にリリースする
-----------	-------------------

Save As: Distribution.p12  Tags: Where: Desktop	①「where」で保存先を指定します。
Enter the password used to secure the Developer Profile Password: Verify:	②確認のため、もう一度同じパスワードを入力します。
Cancel Save	④「Save」をクリックします。

あとはアップロードして待つだけです。

Review AppStore.ipa content:	AppStore.app	
	SUMMARY	
	Team: Tatsuya Tobioka Certificate: iOS Distribution (Expires 2018/10/14)	
	Profile: iOS Team Store Provisioning Profile: com.tnantoka.AppStore © Symbols: Included	
	Bitcode: Included Architectures: arm64	
	ENTITLEMENTS	
	H4Z42X3L6X.com.tantoka.AppStore	
	H4Z42X3L6X.com.tnantoka.AppStore	
Cancel Export	Previous	「Upload」をクリックします。

アップロードが成功すると、次のような画面が表示されます。

Review AppStore.ipa content:		
AppStore.app	AppStore.app	
	SUMMARY Team: Tatsuya Tobloka Certificate: IOS Distribution (Expires 2018/10/14) Profile: IOS Team Store Provisioning Profile: com.tnantoka.AppStore © Symbols: Included Bitcode: Included Architectures: arm64	
Cancel Export	ENTITLEMENTS application-identifier H4Z42X3L6X.com.tnantoka.AppStore keychain-access-groups H4Z42X3L6X.com.tnantoka.AppStore Previous	「Done」をクリックします。

#### Note

「Validate…」というアップロード前にアーカイブを検証する 機能もあります。しかし、たとえ検証の結果問題が見つからなく ても、実際にアップロードするとエラーが発生することもありま す。いきなりアップロードしてしまっても良いでしょう。 また、回線が不安定などの理由でアップロードが失敗する場合

は、Xcodeに含まれる「Application Loader」というアプリを 使うとうまくいくことがあります。「Application Loader」は、 次のようにXcodeのメニューバーから起動することができます。

Xcode File Edit	View Find	d Navigate	Editor	Product
About Xcode		iPhone		
Preferences Behaviors Toolchains	೫, ►	88 < >	No Selectio	n
Xcode Server				
Open Developer Te	Ioc	📕 Instrun	nents	
Services	•	🖬 Simula	tor	
Hide Xcode	жн	Access	sibility Ins	pector
Hide Others	∖сжн	🗊 FileMe	rge	
Show All		🖂 Applica	ation Loa	der

#### iTunes Connect上で確認・設定

アップロードが完了すると、iTunes Connectの「アクティビティ」にアップロードしたアプリが表示されます。



## 注意

「アクティビティ」にアップロードしたアプリが表示されない場合、何かエラーが起きている可能性があり ます。

例えばアイコンの設定を忘れているかもしれません。そのような場合には、Apple社からメールが届くは ずです。アイコンを忘れていた場合は、以下のように対応します。ここではデフォルトのままユニバーサルに してしまっていたので、iPhone専用アプリへの変更も行っています。

#### Chapter 1 App Store CUU-ZJa







アプリをアップロードし直す場合、同じバージョン番号、かつ同じビルド番号ではアップロードすることが できません。ここでは同じバージョンの再提出なのでビルド番号のみを変更します。
▼ Identity			
Display Name	AppStore		
Bundle Identifier	com.tnantoka.AppStore		
Version	1.0		
Build	2	]	「Build」を2に変更します。

# 審査用のビルドを指定

「アクティビティ」画面のアプリ情報から「処理中」という表示がなくなると、審査に進むことができます。 処理が終わったときにも Apple 社からメールが届きます。

	🗑 Apple Inc.	Ċ	1 0 +	
iTunes Connect マイ A	App ~ (例 AppStoreサンプル ~		一 奈岡辰哉 🎽 🛛 ?	
App Store 機能 Tes	stFlight アクティビティ			
iOS 単世 学べてのビルド App Store パージョン 詳愛とレビュー IOS App	iOS ビルド ios 用に提出されたすべてのビルドです。バージー ッパージョン10 ビルド アップロー 5 2 2007年10月	가 한국실 Xcode /( ジョン) E App Stor 48:11:34	部号です。 re ステークス	― 「処理中」の表示がなくなったことを確認し
	Copyright ID 2017 Apole Inc. All rights reserved. 連邦先	プライバシーポリシー		

# アップロードしたファイルを提出用に設定します。

$\bullet \circ \bullet < > \square$	€ Apple Inc. C	ð Ø +
iTunes Connect マイ App	✓ ■ AppStoreサンプル ~	飛翔辰哉~   ⑦
App Store 機能 TestFlig	ght アクティビティ	
APP STORE 供短 App 情報	iOS App 1.0 ● 提出準備中	保存 審査へ提出
価格および配信状況	3987	
iOS APP ● 1.0 提出準備中	> iMessage App	
(+) バージョンまたはブラットフォーム	> Apple Watch	
	ビル h⊕	
	App を送信する前にピルドを選択( ビルドの歴出には、Xcode 6 以風、または Applicati してください。	ってください。 on Loader 3.0 以降を使用



マウスカーソルを表示されたビルドにホバーすると表示される「-」(マイナス)をクリックすると取り消 すことも可能です。

選択したビルドが表示されます。

以上でArchiveの送信は終わりです。

ビルド

S 1.0 (2)

# 5. スクリーンショットなどの登録 ………

アップロード日

2017年10月14日 11:34 🛡

Archiveの送信が済んだら、あとは必要な情報をiTunes Connect上に登録すれば審査に提出することができます。

まず、アプリのスクリーンショットが1~10枚必要です。必要最低限で構わないのでしたら、現行モデル で最も画面サイズが大きい「iPhone 8 Plus」で撮ったスクリーンショットがあれば大丈夫です。



スクリーンショットを取得します。



スクリーンショットは、次の場所にドラッグ&ドロップすることでアップロードすることができます。





他の端末用にも自動でリサイズされて使用されます。

#### Note

各端末のスクリーンショットを個別に指定したい場合は、「メディアマネージャーですべてのサイズを表示」 から追加でアップロードすることも可能です。

# 6. 審査に提出する ……

以上で全ての準備が整いました。最後に審査に必要な情報を入力して提出します。

# 販売価格を設定します。



#### Chapter 1 App Store にリリースする

		i Apple Inc.	<u>a</u> 0
iTunes Connect マイ App	~ (ii) App	oStoreサンプル ~	飛岡辰哉~   ⑦ Tatsuya Tobioka
App Store 機能 TestFli	ght アクティビティ		
APP STORE 情報	価格および配信状	況	
App 情報 価格および配信状況			
IOS APP	価格表		すべての価格と通貨
◎ 1.0 提出準備中	価格 ②	開始日 ③	終了日 ②
(i) K−ジョン≅たはブラットフォーム	JPY 0 (無特) JPY 120 (屈格带 1) JPY 240 (価格带 2) JPY 360 (価格帯 3) JPY 480 (価格帯 5) JPY 600 (価格帯 5) JPY 720 (価格帯 7)	、まず価格を選択してください。	
	JPY 960 (価格帯 8) JPY 1080 (価格帯 9) JPY 1200 (価格帯 9) JPY 1200 (価格帯 10)	1 言可能 ?	



# Note

保存後に価格を変更したい場合は、「価格計画の変更」をクリックします。

「概要」「キーワード」「サポート URL」を入力します。「プロモーション用テキスト」や「マーケティング URL」は任意です。

	₿ Apple Inc.	<u> </u>	+
iTunes Connect マイ App	~ (創 AppStoreサンプル	、 、 、 、 、 、 、 、 、 、 、 、 、 、	
App Store 機能 TestFlig	ht アクティビティ		
APP STORE 情報 App 情報 価格および配債状況	iOS App 1.0 ◎ 提出準備中	保存 憲法へ提出	
iOS APP ● 1.0 提出準備中	プロモーション用テキスト ②	キーワード ② AppStore,サンブル 87	
(+) バージョンまたはプラットフォーム	12	v http://example.com	
	AppStoreのサンブル	マーケティングURL ③ http://example.com (オプション)	各種情報を入力しま
	l		

AppStoreで表示されるアイコンをアップロードします。アプリ内でも指定した 1024pxのアイコンを 「ファイルを選択」と表示されている場所へドラッグ&ドロップすればOKです。

••• <>	🔒 Apple Inc.	¢ å Ø .	
iTunes Connect マイ App	AppStoreサンプル ~	一 飛岡辰哉~ Tatsuva Tobloka	
App Store 機能 TestFlig	ght アクティビティ		
APP STORE 情報 App 情報 価格および配信状況	iOS App 1.0 ◎ 提出準備中	② 信符済み         審査へ提出	
iOS APP ● 1.0 提出準備中 ④ バージョンまたはブラットフォーム	App 一般情報 App Store 7132	Copyright 🛞	
	ファイルを選択 バージョン 参 10		┼── <u> </u> アイコン画像をここへドラッグ&ドロップします。 
	年齢乾限指定 編集 年齢利限指定なし	Tokyo	
●●● <>> □□ iTunes Connect マイ App	● Apple Inc.	さ 一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一	
App Store 機能 TestFlig	ght アクティビティ		
APP STORE 俳句 App 情報 価格および配信状況	iOS App 1.0 ◎ 提出準備中	【 <b>保存</b> 】 [ 憲章へ提出 ]	
los APP ● 10 提出準備中	App 一般情報 App Store アイコン 2	Copyright 👔	
(+) バージョンまたはブラットフォーム	S	<ul> <li>         · · ·</li></ul>	アップロードされました。
	バージョン ⑦ 1.0 年納刻限指定 編集	维 名 Japan v	
	年齢制限指定なし		

Chapter 1 App Store にリリースする

著作権情報(Copyright)と年齢制限指定を入力します。



ここでは「年齢制限指定」はすべて「なし」に設定します。

		Apple Inc.	Ċ		₫ Ø <sub>+</sub>	
iTunes Con	nect マイ App ~ 💮 A	ppStoreサンプ	ル ~	飛I Tatsuya	国辰载~   ⑦	
App Store	生齢期限指定の編集					
APP STORE 情	コンテンツの説明ごとに、App に最も当てはまる 動制限指定は、全プラットフォームで同じ値が表 が基準になります。詳しく見る。	頻度のレベルを選 示されます。最も調	択します。App Store   高い年齢制限指定の Ap	に表示される App 年 ap プラットフォーム	審査へ提出	
価格および	App には、一切の種類のわいせつ、性約、不快、中傷的 たは Apple の論理的な判断によって好ましくないと判難	はコンテンツや資料 される他のコンテン!	(文章、グラフィック、イ ソや資料が含まれていては	「メージ、写真など)、ま はなりません。		
IOS APP	Apple コンテンツの詳細	なし	まれ/軽度	頻繁/極度		
● 1.0 提出半	アニメまたはファンタジーバイオレンス	0				
	リアルな暴力的表現	0		0	路先情報を	
争 バージョン	長時間の過激なまたは加虐的でリアルな暴力	0	0	0		
	冒とく的または下品なユーモア	0		0		[
	成人向けまたは成人向けを暗示するテーマ	0		0		アプリ内でどのようなコンテンツが表示され
	ホラー/恐怖に関するテーマ	0	0	0		などの質問が並んでいます。
	医療/治療情報	0		0		
	アルコール、タバコ、ドラッグの使用または 言及	0		0		
	疑似ギャンプル	0	0	0		
	second states and the second sec	0	0	0		
•		Apple Inc.	Ċ		₫ Ø +	
unes Co	成人向けまたは成人向けを暗示するテーマ		0	0	哉~│ ⑦	
an Store	ホラー/恐怖に関するテーマ	۲	0	0		
10 51010	医療/治療情報	۲	0	0		
P STORE 情報	アルコール、タバコ、ドラッグの使用または 言及	۰			審査へ提出	
価格およ?	疑似ギャンプル	۲	0	0		
	性的内容およびヌード	۲	0	0		
S APP	過激な性的表現およびヌード	۲	0	0		
1.0 提出滞的		いいえ	はい			
15-21	無制限のWebアクセス	۰			絡先情報を	
	ギャンブルおよびコンテスト	۲	0			
	□ 子ども向けに制作	$\cup$				
	<ul> <li>選択した App の年齢制限指定:4+歳。</li> </ul>					

Copyright © 2018 Tatsuya Tobioka / Socym Co., Ltd.

キャンセル 終了

「終了」をクリックします。

「年齢制限指定」が「4+」(4歳以上)と表示されていることを確認し、氏名や連絡先情報を入力します。



入力が済んだら保存します。

	Apple Inc.	Ċ	<u>a</u> 0
iTunes Connect マイ App	<ul> <li>AppStoreサンプル ~</li> </ul>		一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一
App Store 機能 TestFlie	ght アクティビティ		
APP STORE 情報 App 情報 価格および配信状況	iOS App 1.0 ◎ 提出準備中		((RP))
IOS APP	App Review に関する情報		
<ul> <li>● 10 提出準備中</li> <li>サインイン</li> <li>ユーザ名。</li> <li>インでき、 アするに</li> <li>(・) パージョンまたはプラットフォーム</li> </ul>	サインイン情報 ② ユーザ名とパスワードを提供して、App にサイン インできるようにしてください。App Reviewを完 了するには、サインイン情報が必要です。	連絡先情報 ? Tobioka	Tatsuya
	□ サインインが必要です	100000000000000000000000000000000000000	Indevention CCT.
	×=: ?		
			4000
	添付ファイル ②		
	ファイルを選択 (オプション)		

「審査へ提出」をクリックして、提出手続きを行います。アプリの内容に合わせて輸出コンプライアンス、 コンテンツ配信権、広告IDに関する質問に回答してください。

## Chapter 1 App Store にリリースする





••• <>	Apple Inc.	Ċ Č P	①コンテンツ配信権 広告IDに関する質問を確認し
iTunes Connect マイ App	<ul> <li>S AppStoreサンプル~</li> </ul>	田回調證 × ⑦ Tatsuya Tobioka	
App Store 機能 TestFlig	ht アクティビティ		
APP STORE 信服 App 情報		キャンセル 【 送信】	②「送信」をクリックします。
価格および配信状況	コンテンツ配信権		
iOS APP ● 1.0 提出準備中	あなたの App は、サードパーティ製のコンテンツを含む、ii スしますか?	表示する、またはそれにアクセ 〇 はい ⑧ いいえ	
⊕ パージョンまたはブラットフォーム	広告ID (Advertising Identifier又はIDFA)		
	このApp は広志り (DFA) を使用しますか? 広志り (DFA) は、名 (DS デバイスの場所ので、ユーザにタ- かの場ーの方法です。ユーザは (DS デバイス)での サップ ます。 広志り (DFA) の利用に関する正しい間等を選択しているこ DFA が含まれていて「いいいえ」を選択した場合に、パイト の場合、割しいパイナリを提出するの凄かあります。	○ はい ● いいえ ーグティング広告を提供するた ティング広告の制限を選びさ とを確認してください。App に リが個火的に加下されます。こ	
-	Copyright © 2017 Apple Inc. All rights reserved.   サービス規約   プライバ	(シーボリシー   連絡先	

元の画面に戻ります。画面左側のステータス表示が「審査待ち」に変わったことを確認してください。

	B Apple Inc.	¢ 0 +	
iTunes Connect マイ Ap	p ~ S AppStoreサンプル ~	飛岡辰哉 ~ Tatsuya Tobioka	
App Store 機能 TestF	light アクティビティ		
APP STORE 情報 App 情報	iOS App 1.0 ◎ 審査行ち	深存	「南木往と」に亦わったことを確認します
価格および配信状況 IOS APP	バージョン情報	日本語 ~ ②	番重付り]に変わうたことを唯祕します
<ul> <li>● 1.0 署査待ち</li> <li>(十) パージョンまたはブラットフォーム</li> </ul>	App プレビューとスクリーンショット ② Phone 55インチディスプレイ 12.0インチディスプレイ	メディアマネージャーですべてのサイズを表示	
	0/3 App プレビュー   1/5 スクリーンショット   ファイルを週	(沢) すべてを削除	

あとは Apple 社による審査の結果を待つのみです。

# 審査をキャンセルする

審査をキャンセルする場合は、「このバージョンを審査から削除」をクリックします。削除のリンクが表示 されない時は、App情報など他のページにいったん移動したり、ログインし直したりしてみてください。



**Chapter 1** App Store にリリースする





# Note

iTunes Connectの日本語版のヘルプが公開されています。困ったことがあれば参照してください。 『iTunes Connect デベロッパヘルプ』

http://help.apple.com/itunes-connect/developer/?lang=ja



iOSアプリを AppStore で公開するためには、Apple社の審査を通過する必要があります。審査で違反と見なされる事項は Apple社が公開するガイドラインに記載されています。

『App Store審査ガイドライン』

https://developer.apple.com/app-store/review/guidelines/jp/

例えば、次に挙げるような事由でリジェクトされることがあります。

- クラッシュする。
- アプリ名に検索されやすいようにキーワードを入れている。
- 何に使うかわからない情報が表示されている。
- ストレージガイドラインに違反している。(18章で紹介したディレクトリーを適切に使用しなかったなど)
- 広告IDを広告表示に使用すると選択したのに、広告が表示されない。
- 販売されているアプリのテンプレートを元に作成したアプリである。
- iPadのiPhoneモードでちゃんと動かない。

「作ってから実は違反だった」などとならないように、常に最新のガイドラインを確認してください。

ここではアプリをストアにリリースする手順を紹介しました。開発したアプリを App Store に公開する際の参考にしてください。

書籍本体、及び本PDFで紹介したWebサイトの一覧を以下にまとめます。

●本書のサポートサイト
 ●サンプルプログラムのダウンロード

#### 書籍

#### Chapter 1

- Apple Developer Programのサイト
- Apple社の開発者サイト

#### Chapter 2

- Apple Developer Documentation(英語)
- Stack Overflow(英語)
- Apple Developer Forums(英語)
- Qiita(日本語)
- teratail(日本語)

#### Chapter 3

Apple Worldwide Developers Conference 2017資料

#### Chapter 7

● [Stadiometer] ソースコード (ARKit活用例)

# Chapter 8

- [Kickstarter] ソースコード
- Gradientor」ソースコード
- [edhita] ソースコード
- Apple社サンプルコード

#### Chapter 11

● <u>SpriteKit.jp: Sprite Kit 日本語情報サイト</u>

#### Chapter 12

- UIKitの公式ドキュメント(UIButton)
- UIKitの公式ドキュメント(UIControlEvents)

#### Chapter 13

 Apple Human Interface Guidelines/Overview/ iPhone X

## Chapter 15

Core Image Filter Reference

#### ボーナスPDF

- Chapter 1
- Icon Creator
- Apple Human Interface Guidelines
- GitHub Pages
- Netlify
- WordPress.com
- Apple開発者用サイト
- iTunes Connect
- iTunes Connect デベロッパヘルプ
- App Store審査ガイドライン

# Chapter 2

- App Store審査ガイドライン
- Apple技術資料「Technical Q&A QA1689 Supporting orientations for iPad apps」
- Apple Human Interface Guidelines/Visual Design
   Layout

#### Chapter 4

- CocoaPods
- SPM(Swift Package Manager)
- Carthage/README.md
- ●Carthageリリース一覧
- Alamofire
- PKHUD
- Eureka
- SwiftlconFont
- FontAwewome
- KeychainAccess
- Realm
- Realm日本語ドキュメント
- LicensePlist
- ●LicensePlistリリース一覧
- The MIT License
- Apache License, Version 2.0
- LicensePlist Settings.bundle
- SwiftLint
- SwiftLintリリース一覧

# 『Swift 4プログラミング入門』ボーナスPDF

# iPadに対応させる

Chapter 2

iPadは、iPhone専用のアプリでも、拡大表示を して使うことができます。しかし、iPadに対応した アプリの方が使いやすいのは間違いありません。本 PDFの Chapter 2では iPad対応の方法や注意点に ついて紹介します。

Section 2.1	<u>iPad対応をする理由</u>
Section 2.2	iPad 対応プロジェクト
Section 2.3	iPadアプリに必要なアイコン
Section 2.4	Split View Controller
Section 2.5	iPad対応の注意点



App Store 審査ガイドライン(<u>https://developer.apple.com/app-store/review/guidelines/jp/</u>) に以下の記述があります。



iPad対応していないことを理由に審査がリジェクトされる可能性は低いですが、やはり Appleとしては iPadでも動くようにしてほしいという思いがあるようです。また、iPadユーザーにとってもダウンロードし たアプリがiPadにも対応しているとやはり嬉しいものです。

ただ、全てのアプリが iPad 対応の手間をかけられるわけではないですし、そもそも iPad に向かないアプリもあるでしょう。対応の余地があるアプリであれば是非検討してみてください。

アプリの絶対数がiPhoneよりも少ないので、ランキングに入りやすい可能性もあるでしょう。





iPhoneと iPadを大きく見た目を変える場合、かなり手間がかかりますが、そのままの見た目で対応する 場合はオートレイアウトの恩恵もあり、とても簡単です。Storyboardファイルで、デバイスの大きさが変わっ ても適切に表示されるようAutoLayoutを設定しておけば、iPadでもうまく表示させることができます。 Xcode 9以降で新たに作るアプリであれば、デフォルトで iPad対応の Universal(ユニバーサル)ア

プリになっています。

Choose options for	your new p	roject:			
	Pr	oduct Name:	Universal		
		Team:	None	٢	
	Organi	zation Name:	tnantoka		
	Organization Identifier:		com.tnantoka		
	Bun	dle Identifier:	com.tnantoka.Universal	_	
		Language:	Swift	٢	
			Use Core Data Include Unit Tests Include UI Tests		
Cancel				Previous	Next

Xcode 8以前ではデバイス選択欄がありましたが、Xcode 9ではデバイス選択欄 がなくなり、デフォルトで「Universal」になっています。





「iPad Pro (10.5-inch)」を選んでアプリを実行してみます。

t View	Device
A Unive	No devices connected to 'My Mac'
	Build Only Device
	Ceneric iOS Device
	IOS Simulators
	🏢 iPad (5th generation)
t	iPad Air
	iPad Air 2
6	iPad Pro (9.7-inch)
/board	iPad Pro (10.5-inch)
	Pad Pro (12.9-inch)
	Pad Pro (12.9-inch) (2nd generation)
	Phone 5s
	Phone 6
	iPhone 6 Plus
	Phone 6s
	Phone 6s Plus
	Phone 7
	Phone / Plus
	IPhone 8     IDhone 9 Divis
	Dhane CF
	Phone SE
	IPhone X
	Add Additional Simulators
	Download Simulators

これだけでiPad対応のアプリとなります。

Carrier 🇇	1:48 PM	100%
	Universal	

iPhoneだけに対応したい場合には「Deployment Info」の「Deployment Target」を「iPhone」に 変更します。

Deployment Info		
Deployment Targ	iPhone iPad / Universal	─ 「iPhone」を選択します。
Main Interface	Main	
Device Orientation	Portrait Upside Down Landscape Left Landscape Right	
Status Bar Style	Default © Hide status bar Requires full screen	

設定変更後に再度アプリを実行すると、iPhoneモードで実行されます。



# Note

もしiPhone対応が不要であれば、iPadを選べばiPad専用のアプリになります。



iPadアプリの開発には、以下のようなサイズのアイコンが必要です。

サイズ	必須	サイズ	必須	サイズ	必須
20px	_	58px	_	152px	0
29px	-	76рх	0	167рх	0
40px	_	80px	_	1024px	○ (Xcode 9から)

本 PDFの Chapter 1 で紹介した iPhone アプリで必要とされるアイコンサイズと一部重複しています。 ユニバーサルアプリの場合、どちらのサイズのアイコンも必要です。

Assets.xcassetsで iPadのみをチェックした状態にすると、以下のように iPadに必要なアイコンの設 定だけが表示されます。

設定箇所は10箇所ありますが、40pxが2箇所あるため全部で9種類になります。





書籍版の Chapter 1 で紹介した「Master-Detail App」テンプレートを使ったアプリでは UISplitView Controller というコンポーネントが使われます。



「Master-Detail App」テンプレートを使ったアプリは、デバイスに応じて次のように表示が変化します。

デバイス	方向	表示
iPhone	Portrait	一覧から詳細に遷移していく表示。
	Landscape	一覧から詳細に遷移していく表示。
iPhone Plus	Portrait	一覧から詳細に遷移していく表示。
	Landscape	2ペインに別れた表示。
iPad	Portrait	詳細画面に一覧が重なる表示。
	Landscape	2ペインに別れた表示。

# 

# Portrait

Carrier 🗢	2:29 PM				
Edit	Master	+			
2017-10-14 05:28:54 +0000					

# Landscape

Edit	Master	+		
2017-10-14 05:28:54 +0000				

# 

# Portrait

Carrier 🗢	2:29 PM	· · · · · · · · · · · · · · · · · · ·
Edit	Master	+
2017-10-1	4 05:29:47 +0000	
2017-10-1	403.23.47 +0000	

# Landscape

Edit	Master	+	د Detail	
2017-10-14 05:29:47 +0		<b>′</b> +0		
			Detail view content goes here	

# iPad Pro(10.5-inch)の画面 ······

# Portrait

# Landscape

Carrier 🕈 2:31 PM 100% 📖 +	Carrier 🗇 2:31 PM	100% 🔲 🕯
Edit Master + Detail	Edit Master + Detail	
Edit         Master         +         Detail           2017-10-14 05:31:08 +0000	Edit         Master         +         Detail           2017-10-14 05:31:08 +0000         -	

データを一覧表示するようなアプリの場合、「Master-Detail App」テンプレートを使うことで、iPhone、 iPadに対応したユニバーサルアプリを簡単に開発することができるでしょう。

#### Note

例えば、標準アプリの「メール」や「メモ」「設定」のようなアプリを作りたい場合、UISplitView Controllerの使用を検討するとよいでしょう。



アプリをiPad対応するときの注意点です。

# iPhone向けのコードがクラッシュ ……

例えば、書籍版の Chapter 13で紹介したアクションシートのサンプルプログラムは iPad で実行すると クラッシュしてしまいます。

サンプルプログラム Bonus/Chapter2/Popover/Popover/ViewController.swift

```
let alertController = UIAlertController(
    title: "アクションシート",
    message: nil,
    preferredStyle: .actionSheet
)
alertController.addAction(
    UIAlertAction(
        title: "キャンセル",
        style: .cancel,
        handler: nil
    )
)
alertController.addAction(
    UIAlertAction(
        title: "OK",
        style: .default,
        handler: nil
    )
)
present(alertController, animated: true, completion: nil)
```



# エラーでアプリが終了してしまう。

Popover > 📦	IPad Pro (10.5-Inch) Running Popover on IPad Pro (10.5-Inch)			
	😫 < > 📓 Popover > 📄 Popover > 🗟 AppDelegate.swift > No Selection			D Ø
Popover PID 14895         ③ ③           CPU         0%           Memory         44 MB           III         Disk         Zero KB/s           Metwork         Zero KB/s	<pre>// AppDelegste.swift // Popover // Popover // Crested by Tobioka on 2017/10/14. // Copyright = 2017# thantoka. All rights reserved. // // Copyright = 2017# thantoka.</pre>			
Thread 1 Queue: com _thread (serial)	9 import UIKit			
V Thread 1 Gueue: comthread (serial)     Cthread, Sill     11 UkppicationMain     11 UkppicationMain     13 start     14 start     Thread 2     Thread 3     Thread 4     Thread 5     Thread 7     Thread 7     Thread 8	<pre>10 001ApplicationMain 12 class AppDelegate: UIResponder, UIApplicationDelegate { 13 14 15 15 16 17 17 17 18 17 18 17 18 18 17 19 18 17 19 19 19 19 19 19 19 19 19 19 19 19 19</pre>	mishLaunchingWithOptions launchOpt ation launch. om active to inactive state. This phone call or SMS message) or when d state. Le timers, and invalidate graphics abolication d	Thread 1: signal SIGABRT ons: ean occur for certain types of a the user quits the application rendering callbacks. Games	No Selection
	28 // Use this method to release shared resources,	save user data, invalidate timers,	, and store enough application	D {} © D
	<pre>state information to restore your applicatio 29</pre>	n to its current state in case it : tion, this method is called insteam	is terminated later. I of applicationWillTerminate:	Button - intercepts touch events and Button sends an action message to a target object when it's tapped.
	31       32       func applicationWillEnterForeground(_ application: U         Image: Description of the second sec	IApplication) { d 1 〉 🔲 12 main		Item Bar Button Item - Represents an Item on a UIToolbar or UINavigationItem object.
		15 libdyld.dylib start + 1 ) libc++abi.dylib: terminating wir NSException (lidb)	0x000000010ae83d81 th uncaught exception of type	Fixed Space Bar Button Item - Represents a fixed space Item on a UIToolbar object.
🗑 Filter 🗖 🖬 🖾	Auto \$   © ①	All Output \$	ter 🔟 i 🗖 🗖	Button

Chapter 2 iPad に対応させる

原因は、iPadではActionSheetをポップオーバーとして表示させる必要があるためです。

サンプルプログラム Bonus/Chapter2/Popover/Popover/ViewController.swift

```
alertController.popoverPresentationController?.sourceView = sender
alertController.popoverPresentationController?.sourceRect = sender.bounds
alertController.popoverPresentationController?.permittedArrowDirections =
[.up]
```

# iPad ではポップオーバーで動作

Carrier 🌩	2:05 PM	100%
	アラートを表示	
	アクションシート	
	OK	
	OK	

#### Note

ActionSheetをポップオーバーで表示させても、iPhoneでの実行に影響はありません。

# 

iPadアプリは基本的に全画面方向で使えることが望ましいとされていました。

例えば 2010 年に Apple 社が公開した技術資料 『Technical Q&A QA1689 Supporting orientations for iPad apps』 (<u>https://developer.apple.com/library/content/qa/qa1689/</u>index.html) には、次のような記述があります。

It is strongly recommended that your application support all orientations. (全画面方向 をサポートすることを強く推奨します。)

ー方、現行の『ヒューマンインターフェースガイドライン』(<u>https://developer.apple.com/ios/human-interface-guidelines/visual-design/layout/</u>) には iPadに限定した記述は存在しません。しかし、「可能 であれば縦・横両方向サポートすること」という記述があります。

If possible, support both portrait and landscape orientations. (可能であれば縦・横両方の画面方向をサポートしてください。)

現行の iPadはマルチタスク機能をサポートしており、2つのアプリを並べて使うことができます。マルチ タスク機能に対応するためには、全画面方向のサポートが必要です。

Xcode 9で作成するプロジェクトは、デフォルトの状態でマルチタスク機能をサポートする設定になって います。

マルチタスク機能をサポートしない場合は「Deployment Info」の「Requires full screen」をチェックします。

Deployment Info			
Deployment Target	11.0	V	
Devices	Universal	٢	
Main Interface	Main	✓	
Device Orientation	Portrait     Upside Down     Landscape Left     Landscape Right		
Status Bar Style	Default Hide status bar Requires full screen	0	デフォルトでは「Requires full screen」 はオフになっています。

やはり、基本的にはiPadアプリは全画面方向をサポートすべきと考えた方がよいでしょう。 もともと全画面方向に対応していたアプリなら、iPad対応してしまった方が良いという考え方もできます。 また、もともと1方向限定だったアプリを別方向に対応する場合は、崩れがないか入念に確認しましょう。 キーボード・

キーボードに関しても注意しておく必要があります。

iPadではハードウェアキーボードがiPhoneに比べてよく使われるほか、キーボードを移動・分割するモードが提供されています。



文字入力が重要なアプリでは、テストを欠かさないようにしましょう。

この他の箇所も予期せぬ動作をする可能性があります。単純にユニバーサル化しただけのアプリでも、必ず 実機での動作を確認しましょう。

ここでは iPad 対応について簡単に紹介しました。 特に、同じ画面構成で iPad でも動けば便利そうなアプリなら、是非ユニバーサル対応してみてください。

# 『Swift 4プログラミング入門』ボーナスPDF

# Xcodeの詳しい使い方

Chapter 3

書籍版のChapter 1 では数あるXcodeの機能のうち、本書を進めるための最低限のものだけを紹介しました。Xcodeには他にもたくさんの機能があります。 ここでは開発の効率化に役立ちそうなものをピックアップして紹介します。

Section 3.1	Playground
Section 3.2	変数名の一括変更
Section 3.3	便利なキーボードショートカット
Section 3.4	<u>デバッグの方法</u>
Section 3.5	特殊なコメント
Section 3.6	自動テスト
Section 3.7	



今までは iOS アプリのプロジェクトを使ってきましたが、Playground という気軽に Swiftのコードを書いて試せる機能もあります。断片的なコードを試したいときなどに便利です。

Playgroundの作成…

プロジェクトと同じく、Xcode起動時に表示される Welcome to Xcode 画面から作成できます。また、 Xcodeのプロジェクト内に追加することも可能です。

ここではWelcome to Xcode画面から単体のPlaygroundを作成します。

# サンプルファイル

種類	ファイル
Playground	Bonus/Chapter3/Playground/Blank.playground

Xcode 9から以下のようなテンプレートを利用できるようになりました。

テンプレート	概要
Blank	文字列の変数が1つ定義されているだけのシンプルなテンプレート。
Game	SpriteKitを使ったゲームのテンプレート。
Мар	MapKitを使った地図のテンプレート。
Single View	ラベルをもつ単純なビューを表示するテンプレート。



Choose a template for y	our new playground:				
IOS tvOS macOS			🐨 Fi	ter	
Playground					
Blank	Game	Map	Single View		テンプレート一覧が表示されます。
Cancel			Previou	s Next	

# Note

テンプレートー覧画面は、Xcodeのメニューバーから「File」→「New」→「Playground」を選択するこ とでも表示できます。

ここでは「BLANK」テンプレートを選んで、Playgroundを作成します。「Next」をクリックすると保存 場所を尋ねられるので、適当な場所を選び、「Create」をクリックします。



テンプレートの名前は「BLANK」ですが、実際には次のようなプログラムが入力されています。

//:	Playground	-	noun:	а	place	where	people	can	play
impo	ort UIKit								
var	str = "Hell	Lo,	plays	gro	ound"				

Ready   Today at 9:4	8		
😥 < > 🚡 MyPlayground			── プログレスインジケーター
1 //: Playground noun: -a place where people can play			
3 import-UIKit-			
var·str·=·"Hello, playground"		"Helio, playground"	
プログラ』	<b>^編集領域</b>	結果表示領域	

画面上部のプログレスインジケーターを確認して、シミュレーターが起動する(「Ready」と表示される) までしばらく待ちます。

シミュレーターが起動すると、画面右側に "Hello, playground"という変数 strの中身が表示されます。 以降もコードを編集たびに自動的に表示が更新されていきます。

print .....

これまでのプロジェクトと同じく printを使うと「Debug Area(デバッグエリア)」で出力結果を確認 できます。入力されているプログラムの末尾に、次のプログラムを追加します。すると、画面下部に「Debug Area」が表示されます。

# サンプルプログラム Bonus/Chapter3/Playground/Blank.playground/Contents.swift

print(str)

•	Ready   Today at 10:00		
88	< > 🗟 MyPlayground		
1 2 3 4	//: Playground noun: a place where people can play- import UIKit- ~	"Helio, playground"	=
7	print(str).	"Hello, playground\n"	
▼	•		
Hell	o, playground		

liveViewという機能を使うと、「Assistant Editor(アシスタントエディター)」でビューを確認することができます。

入力されているプログラムをすべて削除して、次のプログラムを入力してください。このプログラムでは単 色のUIViewを表示しています。

### サンプルプログラム Bonus/Chapter3/Playground/Blank.playground/Contents.swift

```
import UIKit
import PlaygroundSupport
// グレーのビューを作成
let view = UIView(
    frame: CGRect(x: 0.0, y: 0.0, width: 320.0, height: 240.0)
```

```
)
view.backgroundColor = .lightGray
// ラベルを作成してビューに追加
let label = UILabel()
label.text = "Hello World!"
label.sizeToFit()
label.center = view.center
label.textColor = .black
view.addSubview(label)
// ビューを表示
PlaygroundPage.current.liveView = view
```

サンプルプログラムを入力後、メニューバーから「View」→「Assistant Editor」→「ShowAssistant Editor」を選択するとAssistant Editorが表示されます。



GameやMap、Singe Viewテンプレートを使う場合には、基本的にliveViewを使うことになります。

Chapter 3 Xcodeの詳しい使い方



[Game]	テンプレー	トの実行画面

O Running Ma	p		
8 < > ≥ Map		③ < > ② Live View : Mep.playground (Live View)	+ >
$\label{eq:response} \begin{array}{  c   } \hline \hline \\ $	_C CLI.costor.Condinast 20 _C CLI.costor.Condinast 20 _C VMCcondinus Region = _C VMCcondinus Region = _C VMCcondinus Region = _C VMCcondinus Region =		
// All terms are any any any and a second an	AMN windowestains: 0x00 AMN windowestains: 0x00 AMN windowestains: 0x00 AMN windowestains: 0x00 AMN windowestains: 0x00 AMN daylow		

「Map」テンプレートの実行画面

• • •	C Running Singlethew	Running Singlet/lew					
8 < > 🖻 Singlet/levr			S < > ⊕ Live View >	+ :			
<pre>/// reserve the size metalling and the meaning out (here limit dot) the reserve the size of the size of the size of the size (the size of the siz</pre>	<ul> <li>δ. μαζετι 2.</li> <li>δ.</li></ul>	8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8		Hello Wond			

「Single View」テンプレートの実行画面

# Project Navigator------

通常の Xcode プロジェクトと同じく、Playground でも「Project Navigator」を表示させることができます。ファイルや画像を追加する場合は「Project Navigator」を使います。

.lightGray		Show Navigator	<b>%</b> 0	
		Show Report Navigator	ж9	
Enter Full Screen	^₩F	Show Breakpoint Navigator	ж8	
Show All Tabs	☆ ₩ \	Show Debug Navigator	¥7	
Show Tab Bar		Show Test Navigator	#6	
Hide Toolbar	∖сжт	Show Issue Navigator	#64 925	
Utilities		Snow Sympol Navigator	#3 @4	
Debug Area		Show Source Control Navigator	¥2	「Show Froject Navigator」を選択しま
Navigators		Show Project Navigator	Ж1	メーユーバーから   View」→   Navigator
Version Editor			_	d = -dS + bS - DG + dA - DH + A - DH
Assistant Editor				
Standard Editor		💥 Running Blank		
View Find Navig	gate Ed	itor Product Debug Source C	ontrol	



例えば独自のクラスをPlaygroundで使いたい場合は、「Project Navigator」の「Sources」に.swift ファイルを追加し、その中にクラスを書くことで実現できます。ただし、publicか openなクラスとして定 義しないとPlaygroundから呼び出せないため注意してください。

また、画像などを使いたい場合は「Resouces」に追加します。その画像はUIImage(named:)などの方法で読み込むことができます。

このようにビルドしたり、シミュレーターを起動したりすることなく、書いたプログラムを手軽に試すこと ができます。

# Note

書籍版のPart 2で掲載しているサンプルプログラムはPlaygourndでも動作を検証しています。 サンプルプログラムのPlaygroundフォルダーの中に入っていますので、必要に応じてご利用ください。



Xcodeには変数名などの変更を一度に行えるリファクタリング機能がついています。複数ファイルをプレビューしながら行えるので便利です。

# サンプルファイル

種類	ファイル	
プロジェクト	Bonus/Chapter3/Refactor.xcodeproj	
Storyboard	ryboard 変更しません	
ソース	Bonus/Chapter3/Refactor/Refactor/ViewController.swift	

ViewController.swift内に、定数を1つprintするコードを書きます。

サンプルプログラム Bonus/Chapter3/Refactor/Refactor/ViewController.swift

let before = "
\$\cap\$+\$\sigma + \$\cap\$"
print(before)

続いて、別のファイルから、この定数を使います。

サンプルプログラム Bonus/Chapter3/Refactor/Refactor/AppDelegate.swift

print(ViewController().before)

それでは、リファクタリング機能を使って定数名を変更してみます。

13	/// [marker1]	①変更したい定数 (ここでは View Controller.	
14	let befor - "=+", t "		
15	/// [mar Conv Symbol Namo		SWIITの方)を選択して、右クリックします。
17	override Paste	<b>•</b>	
18 19	super // Refactor	Rename	
20 21 22 23 24	/// Find Selected Text in Workspace prin Find Selected Symbol in Workspace Find Call Hierarchy	Extract Function Extract Method Extract Variable Extract All Occurrences	→ [Rename] を選択します。
25 26 27	Show Issue Add Missing Abstract Class Jump to Definition Add Missing Protocol Requ	Add Missing Abstract Class Overrides Add Missing Protocol Requirements	
28 29	} Show Blame for Line	Add Missing Switch Cases	
30	Open in Assistant Editor	Expand Default	
All (3) Code (3) File Names (0) Comments (0) Other (0)	Cancel		
--	--------		
✓ ▼ ⊇ ViewController.swift			
13 /// [marker1] 14 let before = "テキスト" 10 /// [morker1]			
21         /// [marker2]           22         print(before)           23         /// [marker2]	]		
🗹 🔻 🔄 AppDelegate.swift			
20     /// [marker1]       21     print(ViewController() before)       22     /// [marker1]			

選択した定数を使用しているファイルが開き、定数が使われている箇所が表示される。

All (3) Code (3) File Names (0) Comments (0) Other (0)	Cancel	ame
ViewController.swift		
13 /// [marker1] 14 let after] = "テキスト" 10 /// UNGINET1]		
21 /// [marker2] 22 print(after) 23 /// [marker2]		
V AppDelegate.swift		
20     /// [markar1]       21     print(ViewController()_after)       22     /// [marker1]		
<ol> <li>「after」に変更すると、該当箇所がすべて自動的に変</li> </ol>	更されます。	
	 ②右上の「Rename」ボタンをクリックする	L ると変

このように一括でプレビューしながら変更できます。

# Note

リファクタリング機能は Objective-C向けとしてはバージョン 8以前の Xcodeでも提供されていました。 Swiftには長らく対応していませんでしたが Xcode 9でついに対応されました。とても便利な機能ですので、 ぜひ活用してください。



Apple社は Xcode Keyboard Shortcuts and Gestures という文書を公開しています。この文書には Xcode で利用できるショートカットが多数記載されていますが、覚えきれる量ではありません。また、この 文書は2011年から更新されていません。ここでは有用と思われるショートカットを厳選して紹介します。 以降で紹介するショートカットは、2018年3月にXcode 9.0での動作を確認しています。

#### Xcode ·····

操作	+-
実行	command + R
ビルド	command + B
終了	command +.
テスト	command + U
コメントアウト・アンコメント	command + /
クイックオープン	command + shift + 0
矩形選択	option +マウス
行の先頭にカーソル移動	control + A
行の末尾にカーソル移動	control + E
行選択	shift+↑/shift+↓
検索	command + F
自動インデント整理	control + I
アシスタントエディターに切り替え	option + command + return
標準エディターに戻す	command + return

ビルドはアプリの実行はせずにソースコードのコンパイルのみ行う機能です。

**クイックオープン**は、以下のようにファイル名の部分一致などを元に、素早く目的のソースコードなどを開くことができる機能です。



**矩形選択**は複数行の同じ列を一度にコピー&ペーストしたりできる機能です。

```
18 // ラベルを作成してビューに追加
19 let label = UILabel()
20 label.text = "Hello World!"
21 label.sizeToFit()
22 label.center = view.center
23 label.textColor = .black
24 view.addSubview(label)
```

# 

操作	+
デバイス回転	command + - / command + -
ホームボタン	command + shift + H
スクリーンショット	command + S
ハードウェアキーボード有効・無効切り替え	command + shift + K
ソフトウェアキーボード表示・非表示切り替え	command + shift + K

ここで紹介したショートカットキーだけでも使いこなせれば、作業効率が向上するはずです。



プログラムがうまく動かない時に、printを使って変数の中身などを確認することがあります。この手法 は printデバッグと呼ばれ、手軽に使えて便利ですが、コードに変更を加えているためあまり良い方法とはい えません。

理論的には printを追加したことでプログラムの処理が変わってしまい、確認したいバグが再現(発生) しなくなってしまう可能性もあります。また、printを消し忘れてデバイスのログに大事な情報を出力し続 けてしまう可能性もあります。

#### サンプルファイル

種類	ファイル
プロジェクト	Bonus/Chapter3/Debug.xcodeproj
Storyboard	Bonus/Chapter3/Debug/Debug/Base.lproj/Main.storyboard
ソース	Bonus/Chapter3/Debug/Debug/ViewController.swift

# 

Xcodeには高機能なデバッガーが付属しているため、実行中のプログラムで使われている変数の値を見るのは簡単です。

ここでは、筆者が用意したサンプルプログラム(画面をタッチしたら count変数をインクリメントするアプリ)を使ってデバッグやログ出力の方法を説明します。

#### プログラムの編集時に行番号を表示させる

Editorでサンプルプログラムを開き、行番号が表示されていない場合は、Xcodeのメニューバーから 「Xcode」→「Preferences」を選択し、「Text Editing」画面の「Editing」タブで「Show:」グルー プの「Line Numbers」をオンにします。行番号がなくても操作はできますが、わかりづらいので表示して おく方がよいでしょう。

General Accounts Behaviors Navigation Fonts	Text Ediling	
Code completion While editing Default text encoding	Editing     Indentation       Page guide at column:     80       If ghilight instances of selected symbol       Delay:     0.25       Suggest completions while typing       Use Escape key to show completion suggestions       Automaticity insert closing braces (7)       Enable type-over completions       Automaticity balance brackets in Objective-C method calls       Enclose selection in matching delimiters       Automaticity trim trailing whitespace       Including whitespace-only lines       Uncode (UT-8)	「Line Numbers」をオンにします。
Default line endings Code coverage	macOS / Unix (LF)     Convert existing files on save     Show iteration counts	

# 

行番号部分をクリックするとブレークポイントが設定されます。

ここではViewController.swiftの25行目をクリックしてブレークポイントを設定します。



色がついた行番号をダブルクリックすると、編集画面が開きます。「Add Action」をクリックします。



Chapter 3 Xcodeの詳しい使い方

count変数の中身をログに出力します。

ViewCont	roller.swift:25		
Condition			
Ignore	0 🗘 times before stopping		
Action	Log Message	+ -	① [Log Message]を選択しま
	@count@		② 「@count@」と入力します。
	Log message to console	@exp@ = expression	
	Speak message	%B = breakpoint name	
		%H = breakpoint hit count	
Options	Automatically continue after ev	aluating actions	

この状態でアプリを実行すると、変数の内容がDebug AreaのConsoleへ出力されます。

9		00	$\rightarrow$	⊥   ©	00	🛷   💮 Debug	-			_	
							0			 C	ount変数の内容が
							2			夛	表示されます。
A	uto 🗘 🗎					Filter	All Output \$	Filter	1		

ブレークポイントは行番号のクリックで無効・有効を切り替えられます。また、行番号部分の外側にドラッ グ&ドロップすると削除できます。



少し難しそうに見えますが、慣れれば簡単にできるようになります。

ブレークポイントを設定することで、プログラムを一時停止して対話的に操作することもできます。 ブレークポイントを設定し、何もせずにそのままアプリを実行してみましょう。

$\frac{2}{2}$	24	override count	func touchesB : += 1	egan(_ to	uches:	Set <uitouch>,</uitouch>	with	event:	UIEvent?)	{
行	8 番号を:	<u>}</u> クリック <sup>-</sup>	するだけです。							

すると、アプリがそこでストップし、「Debug Area」が表示されます。「Debug Area」の左には変数な どが表示されており、中身を確認することができます。

また、「Console」入力可能な状態になっていて、ここで「po 変数名」と入力すると変数の内容が表示されます。変数名の部分はSwiftのプログラムなので、例えば変換して表示するなどもできます。画面の例では countに+ 1しています。

24 override func	touchesBegan( touches: Set <uito< th=""><th>uch&gt;, with event: UIEvent?) {</th><th></th><th>1</th></uito<>	uch>, with event: UIEvent?) {		1
25 count += 1			Thread 1: breakpoint 1.1	
26 }				
27 }				
28				
29				
				- 赤粉めるの中白が主子されます
				愛知やての中身が衣小されまり
	1 0 % √ 0 Debug)	Thread 1 > 🛄 0 ViewController.touchesBegan(_:w	/ith:)	
touches (Set <uitouch>)</uitouch>	)	(11db) po count		
event = (UIEvent?) 0x00	0006040001177c0	0		
ViewContr	oller) 0x00007fe38f105b60	(11db) po count + 1		
▶ UIKit.UIViewController	(UIViewController)	1		
count = (int) 0		(*** ** *		
Auto 🗘   💿 🕕	( Filter	All Output \$	iter 🕼 🛙 🖬 🖬	

「Continue program execution」をクリックするとアプリの処理が復帰します。



デバッガーには他にも様々な機能があります。

まずはprintの代わりに使うことから始めて、少しずつその他の機能も試してみてください。



書籍版の Section 2.4「コメント」で、「コメントの書式に沿って書かれた箇所はプログラムの実行時に無 視されるため、処理に影響を与えることはありません」と説明しました。しかし、Xcodeを使って開発する ときだけ特殊な意味を持つコメントがあります。

こうした特殊なコメントを使うことで、Xcodeがより便利になります。

#### サンプルファイル

作りかけのカウンターアプリのプログラムを例として使います。

種類	ファイル
プロジェクト	Bonus/Chapter3/Comment.xcodeproj
Storyboard	Bonus/Chapter3/Comment/Comment/Base.lproj/Main.storyboard
ソース	Bonus/Chapter3/Comment/Comment/ViewController.swift

Xcodeで使える特殊なコメント表記には、以下のようなものがあります。

表記	意味
MARK	プログラムの区切りなどに印をつける
TODO	積み残しの課題を記載する
FIXME	修正箇所を明示する

カウンターアプリのViewController.swiftを開きます。

まず、カウント用の変数が定義されていることが分かります。

サンプルプログラム Bonus/Chapter3/Comment/Comment/ViewController.swift

var count = 0

続きを読み進めていくと、次の部分で特殊なコメントを使っていることが分かります。

#### サンプルプログラム Bonus/Chapter3/Comment/Comment/ViewController.swift

```
// MARK: - カウンター関連処理
func increment() {
   count += 1
}
func decrement() {
   // FIXME: -=を使う
   count = count - 1
}
func reset() {
   // TODO: リセット
}
```

このように書いておけば、Xcodeが特殊なコメントを認識して表示してくれます。



MARK: -のように - をつけることで区切り線が入ります。適切にコメントを入れておくと、後から編集しやすくなりますので、利用してみてください。



アプリをテストする一番単純な方法は、手動・目視での確認です。しかし、この方法は手間がかかるという 難点があります。

例えば、アルファベットの小文字で入力した文字列を、大文字に変換するアプリがあると仮定します。



最初は単純に大文字に変換するだけでしたが、文字列が未入力の場合にはエラーメッセージを表示したくなりました。すると、以下の両方を確認しなくてはなりません。

#### 元々実装していた大文字に変換する処理

● 新たに実装したエラー表示の処理

このような時に自動テストが役に立ちます。大文字変換の機能を確認する自動テストが既にあれば、それを 実行するだけで元々あった仕様の確認は完了です。あとはエラーメッセージ表示用の自動テストも追加するか、 そこだけ目視確認するなどで確認完了です。

自動テストを書く ……

例にあげた大文字返還アプリをサンプルとして、テストを書いてみましょう。

#### サンプルファイル

種類	ファイル
プロジェクト	Bonus/Chapter3/Converter.xcodeproj
Storyboard	Bonus/Chapter3/Converter/Converter/Base.lproj/Main.storyboard
ソース	Bonus/Chapter3/Converter/Converter/ViewController.swift

# 1. アプリを用意する

入力した文字を大文字に変換する簡単なアプリです。

プ新規Xcodeプロジェクトのオプション選択画面で「Include Unit Tests」をオンにすることで、ロジェ クト作成時にテストを一緒に作成することができます。

Choose options for your new project:			]
Product Name:	Converter		
Team:	None	\$	
Organization Name:	tnantoka		
Organization Identifier:	com.tnantoka		
Bundle Identifier:	com.tnantoka.Converter		
Language:	Swift	0	
	Use Core Data Conclude Uni Tests Include UI Tests		└ [Include Unit Tests]をオンにします。
Cancel		Previous Next	

テキストフィールド、ラベルをOutletで用意しておきます。

```
サンプルプログラム Bonus/Chapter3/Converter/Converter/ViewController.swift
```

```
@IBOutlet weak var textField: UITextField!
@IBOutlet weak var label: UILabel!
```

ボタンから呼ばれるonTapアクションは以下のような単純な処理です。

```
サンプルプログラム Bonus/Chapter3/Converter/Converter/ViewController.swift
if let text = textField.text {
    label.text = text.uppercased()
}
```

#### 2. テストを書く

ConverterTests.swiftファイルにテストを追加します。Buttonのアクションに設定された onTapメ ソッドが呼ばれた時に、テキストが大文字に変換されているかを確認しています。

サンプルプログラム Bonus/Chapter3/Converter/ConverterTests/ConverterTests.swift

```
func testConvert() {
    // Storyboardファイルからコントローラーを取得
    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    let controller = storyboard.instantiateInitialViewController() as!
ViewController
    // ビューにアクセスすることで、ビューを読み込む
    _ = controller.view
    // 変換テスト
    // テキストを設定
    controller.textField.text = "text"
    // ボタンのタップと同じことをする
    controller.onTap(self)
```

}

本来このようなロジック部分はモデルファイルなどを作って、コントローラーとは別ファイルに記載すべき と思われますが、ここでは簡略化してコントローラー内に実装してそのままテストします。

なお、テストでは予期せぬ nilが入ってきた場合などに早期にクラッシュする方が望ましいことも多いので、!を気にせず使います。

#### Note

テストファイルを分けたい場合は「Unit Test Case」を追加します。 「UI Test Case」と間違えないように注意してください。 UIの方はシミュレーターを自動操作して行なうテストです。

// テキストが "TEXT" になっているか確認

XCTAssertEqual(controller.label.text!, "TEXT")

#### 3. テストを実行する

ショートカット command + Uでテストを実行することができます。緑のチェックマークが成功という意味です。



#### 4. 仕様変更

テキストが未入力の場合、エラーを表示する仕様変更を行います。

```
サンプルプログラム Bonus/Chapter3/Converter/Converter/ViewController.swift

if let text = textField.text {

    if text.isEmpty {

        label.text = "テキストを入力してください"

    } else {

        label.text = text.uppercased()

    }

}
```

この状態で再度テストを実行(command + U)すると問題なく通過しました。仕様の変更によって、既存の処理がおかしくなってないことはこれで確認でき一安心です。後はエラーメッセージの処理を確認すれば仕様変更は完了です。

# 

アプリの仕様変更以外にも、リファクタリングや、Swiftのバージョンアップへの対応などで、プログラムの変更が必要となります。

筆者は自動テストの一番のありがたみはリファクタリングがしやすくなることだと考えています。プログラ ムをより気軽に直せるというのは、精神衛生上とてもいいものです。

例えば、先程の処理をguardで書き換えました。

```
サンプルプログラム Bonus/Chapter3/Converter/Converter/ViewController.swift
```

```
guard let text = textField.text else { return }
guard text.isEmpty else {
    label.text = "テキストを入力してください"
    return
}
label.text = text.uppercased()
```

単純な変更であり、テストがないと確認が面倒で飛ばしてしまいそうになります。しかし、実は条件が逆に なってしまっています。テストを実行するとすぐ気づきます。

٩	f	unc testConvert() {
37		let storyboard = UIStoryboard(name: "Main", bundle: nil)
38		let controller = storyboard.instantiateInitialViewController() as! ViewController
39		
40		_ = controller.view
41		
42		<pre>controller.textField.text = "text"</pre>
43		controller.onTap(self)
44		XCTAssertEqual(controller.label.text!, "TEXT") 🛛 🐼 XCTAssertEqual failed: ("テキストを入力してください") is not equal to ("TEXT") -
45	}	
46	}	

正しくは次の通りです。

サンプルプログラム Bonus/Chapter3/Converter/Converter/ViewController.swift

```
guard let text = textField.text else { return }
guard !text.isEmpty else {
    label.text = "テキストを入力してください"
    return
}
label.text = text.uppercased()
```

# これならテストも通りました。

٢	fu	nc testConvert() {
37		let storyboard = UIStoryboard(name: "Main", bundle: nil)
38		let controller = storyboard.instantiateInitialViewController() as! ViewController
39		
40		_ = controller.view
41		
42		<pre>controller.textField.text = "text"</pre>
43		controller.onTap(self)
44		XCTAssertEqual(controller.label.text!, "TEXT")
45	}	
46	}	

全ての機能に対してテストを書くのはとても大変ですが、大事な処理だけでも書いてあると安心です。

#### Note

どこまでテストを書くのか、というのは難しい問題です。

特にiOSは1年に1回大幅なアップデートが行われることもあり、書いたテストが無駄になってしまう可能性 も考えられます。

筆者は「このアプリにとってこの機能がなければ絶対困る」という部分は少なくともテストを書くべきだと 考えています。

例えばメモ帳アプリであれば「メモを作成・変更して保存、それを表示できる」というのはなくてはならな いものです。一方フォントの変更ができるという機能は、必須ではないと考えることもできるでしょう。



Carrier 🗢	4:41 PM	· *	Carrier 🗢	4:42 PM	
	$\frown$				
	Hello			こんにちは	
****		<b>+</b> _			+
央語 (	き1丁した場合の	衣小。	日本語	じまけした場合の	衣小。

# 

種類	ファイル
プロジェクト	Bonus/Chapter3/Localization.xcodeproj
Storyboard	Bonus/Chapter3/Localization/Localization/Base.lproj/Main.storyboard
ソース	Bonus/Chapter3/Localization/Localization/ViewController.swift

# 

アプリの利用者は日本に住んでいる人より、英語圏に住んでいる人の方が圧倒的に多いです。絶対に日本語 でしか使われないというアプリでない限り、英語対応しておくことをお勧めします。そうすることで沢山の人 に使ってもらえる可能性が広がります。

多言語対応はローカリゼーションや国際化(インターナショナリゼーション・i18n)とも呼ばれます。 Xcodeでの多言語対応はとても簡単です。

今まではソースコードに直接日本語で文字列を書いてきましたが、多言語化する場合は、英語を基準として 日本語化するイメージになります。また、Storyboard上でも対応可能ですが、保守が大変なのでコード上で 指定することをおすすめします。

# 事前準備:アプリを用意する ……………………………………………………

ここでは、例としてラベルのテキストに英語では「Hello」と、日本語では「こんにちは」と表示させてみます。

Storyboardの上下左右中央にラベルを配置し、「Assistant Editor」で「label」として Outlet接続しておきます。また、ラベルのテキストと「Hello」と書き換えておきます。

# 

言語情報を管理するファイルを作ります。

Localization	Show in Finder	──_① [Localization] (プロジェクト名)を右クリックします。
AppDelegat	Open with External Editor	
ViewContro	Open As	
💽 Main.storyb	Show File Inspector	
Assets.xca		
LaunchScr	New File	——②「New File」を選択します。
Info.plist	Add Files to "Localization"	







# 

作成されたファイルは多言語化の対象になっていないため、「File inspector」からLocalizeします。

Identity and Ty	ре		① 「Navigator Area」で「Localizable.strings」
Name	Localizable.strings		を選択した状態で「File inspector」を開きます。
Туре	Default - Localizable Strin		
Location	Relative to Group	٢	
	Localizable.strings	iin.	
Full Path	/Users/tobioka/Documents/		
	Chapter23/Localization/		
	Localizable.strings	0	
On Demand Re	source Tags		
Tags			
Localization			
	Localize		-②「Localize…」をクリックします。
Identity and Ty	pe		
Name	Localizable.strings		
Туре	Default - Localizable Strin	$\Diamond$	
Location	Relative to Group	$\Diamond$	
	Containing directory	iin (	
Full Path	/Users/tobioka/Documents/		
	Chapter23/Localization/		
	Localization	0	
Dev Region	/Users/tobioka/Documents/		
	Chapter23/Localization/		
	Localization/en.lproj/	~	
	Localizable.striligs	~	
Localization			
Base		_	
🔽 👔 English			「English」にチェックが入っています

英語用の情報として以下のように記述します。「Hello」を「Hello」に翻訳するという意味になります。(英 語なのでそのままです。)

サンプルプログラム Bonus/Chapter3/Localization/Localization/en.lproj/Localizable.strings

"Hello" = "Hello";

# 3. 日本語の追加 ··

# 言語として日本語を追加します。



「Navigator Area」を見ると日本語用のLocalizable.stringsが作成されていることを確認できるので、 これを編集して日本語の言語情報を記載します。



「Hello」が「こんにちは」に翻訳されるように記述します。

サンプルプログラム Bonus/Chapter3/Localization/Localization/ja.lproj/Localizable.strings

"Hello" = "こんにちは";

# 

NSLocalizedStringを使って、多言語化した "Hello" を呼び出します。commentには空文字列を指定 しておけば OK です。

```
サンプルプログラム Bonus/Chapter3/Localization/Localization/ViewController.swift
```

label.text = NSLocalizedString("Hello", comment: "")

言語を切り替えて多言語化されたテキストの表示を確認します。







実行すると「こんにちは」と表示されます。

#### Note

この手順で毎回設定を変えるのは大変です。

基本は日本語に設定にしておいて、英語を確認したいときはシミュレーターのホーム画面からアプリを起動 すると便利です。

Xcode 経由でアプリを実行しなければ、Scheme メニューで行った設定は反映されないため、ホーム画面から起動した場合はシミュレーターの言語(デフォルト設定だと英語)が使われます。

# 

アプリの表示名は、「Identity」の「Display Name」から変更可能です。

	General	Capabilities	Resource Tags	i Info	Build Settings	Build Phas	es Build Rules	
PROJECT		▼ Identity						
Localization		,						
TARGETS			Display Name	Localization				 ― アプリの表示名
A Localization			Bundle Identifier	com.tnantoka.L	ocalization			
			Version	1.0				
			Build	1				

しかし、言語ごとに表示されるアプリ名を変更したい場合はローカライズが必要です。Localizable. stringsと同じようにInfoPlist.stringsを作成し、以下の手順で多言語化します。

Identity and Ty	pe
Name	InfoPlist.strings
Туре	Default - Localizable Strin 🗘
Location	Relative to Group
	InfoPlist.strings
Full Path	/Users/tobioka/Documents/
	Chapter23/Localization/
	Localization/InfoPlist.strings
On Demand Re	source Tags
Tags	
Localization	
	Localize



Identity and Ty	pe	
Name	InfoPlist.strings	
Туре	Default - Localizable Strin	🗘
Location	Relative to Group	\$
	ja.lproj/InfoPlist.strings	-
Full Path	/Users/tobioka/Documents	5/
	Chapter23/Localization/ Localization/ja.lproj/ InfoPlist.strings	C
On Demand Res	source Tags	
Tags		
Localization		
🔽 🍞 English		
🔽 👔 Japanes	se	

Chapter 3 Xcodeの詳しい使い方

英語のアプリ名を設定します。

サンプルプログラム Bonus/Chapter3/Localization/Localization/en.lproj/InfoPlist.strings

CFBundleDisplayName = "Hello";

日本語のアプリ名も設定します。

サンプルプログラム Bonus/Chapter3/Localization/Localization/ja.lproj/InfoPlist.strings

CFBundleDisplayName = "こんにちは";

実機のiPhoneと同じ手順で、シミュレーターの「設定(Setteings)」アプリから「一般」→「言語と地域」 →「iPhoneの使用言語」を日本語に変えると、アプリ名が日本語になります。(Schemeメニューから変え る方法だと反映されません。)



ここではXcodeの様々な使い方を紹介しました。 気になる機能があればぜひ開発に取り入れてみてください。

# 『Swift 4プログラミング入門』ボーナスPDF

# 外部ライブラリを使う

Chapter **4** 

本格的なアプリを作ろうとすると、すべての機能を 1から作るのは現実的でなくなってきます。幸いなこ とに現在はたくさんのライブラリがオープンソースソ フトウェアとして公開されています。それらのライブ ラリを有効活用させてもらい、開発を効率化しましょ う。

Section 4.1	ライブラリ依存管理ツール
Section 4.2	Carthageの使い方
Section 4.3	ローディング—PKHUD
Section 4.4	フォーム作成—Eureka
Section 4.5	アイコンフォント—SwiftIconFont
Section 4.6	機密情報保存—KeychainAccess
Section 4.7	データベース—Realm
Section 4.8	ライセンスの表示—LicensePlist
Section 4.9	ソースコードのスタイルチェック



ライブラリのソースコードをコピーしてきてプロジェクトに取り込めば、他の人が書いたコードを使う ことはできます。しかし、その方法だとバージョンアップやライブラリの依存関係の解決などが大変にな ります。依存管理ツールを使えば、バージョンや依存関係を管理してくれます。

#### Note

依存関係とは、例えば「ライブラリ Aを使うためにはライブラリ Bが必要で、さらにライブラリ Bを使うためにはライブラリCが必要」というような関係です。これを手動で管理するのは大変です。

Swiftに使える代表的な依存管理ツールは以下の3つです。

- CocoaPods
- Swift Package Manager
- Carthage

# CocoaPods ······

	i coccoapods.org	ð @ +	
(COCO4PODS)	APP ABOUT	GUIDES BLOG	
			[CocoaPods]
			https://cocoapods.org
		0	
JEARCH		<u> </u>	
* Type here to search by name, version, author, keywor	ds, summary, and dependencies.		
WHAT IS	COCOAPODS		

CocoaPodsは Objective-C(Swift 登場以前から iOS開発に使われてきた言語)の時代から使われてい る定番のライブラリ依存管理ツールです。簡単なコマンドを実行するだけで、依存管理に関する様々な仕事を 簡略化してくれる高機能なツールです。CocoaPodsに対応しているライブラリの情報が一元管理されており、 公式サイトで検索できることも魅力です。

一方、様々なことを自動で行なうため、CocoaPodsが原因でエラーになった場合、解決に手間取ること があります。また、Rubyというスクリプト言語で作られているので、Rubyの経験がないと戸惑う可能性も あります。

筆者は数年間お世話になっていますが、予期せぬエラーが発生した時のサポートが書籍では難しいこともあ るため、今回は採用しないことにします。

# SPM (Swift Package Manager) ------

••• <>	Apple Inc.	
Swift	Package Manager	[SPM (Swift Package Manager)]
BLOG		mips.// swm.org/ package-manager/
DOWNLOAD	The Swift Package Manager is a tool for managing the distribution of Swift code. It's integrated with the Swift build system to automate the process of downloading, compiling	
GETTING STARTED	and linking dependencies.	
DOCUMENTATION	The Package Manager is included in Swift 3.0 and above	
MIGRATING TO SWIFT 4	The Fackage Manager is included in Switco.co and above.	
SOURCE CODE		
COMMUNITY	Concentual Overview	
CONTRIBUTING		
CONTINUOUS	This section describes the basic concepts that motivate the functionality of the Swift Package Manager.	
SOURCE COMPATIBILITY		
FOCUS AREAS	Modules	
ABI STABILITY	Swift organizes code into modules. Each module specifies a namespace and enforces access controls on which parts of that code can be used outside of the module.	

SPMは Apple公式のツールです。現在はコマンドラインツールや Linux で使われていますが、iOS でもいずれこちらが標準になっていくことでしょう。現状ではまだ iOSに使う環境が整っていないため、本書では使用しません。





**Carthage**は、GitHub社が2014年に公開した、比較的新しいライブラリ依存管理ツールです。Swift で作られており、pkgファイルも提供されているのでインストールが簡単です。

CocoaPodsと違い、最低限の管理しか行いません。そのため手作業での設定が多くなりますが、 Carthageが原因で問題が起こる可能性が低くなっています。また、CocoaPodsよりもビルドの時間が短 く済むという特徴もあるため、使用するライブラリが増えてくるとその点で優位です。

インストールの簡単さや問題が起こりにくい点を重視して、本書ではCarthageを使用します。

#### Note

Carthageは英語では「カッセージ」のような読み方になりますが、日本では「カルタゴ」と読まれることもあります。



Carthageを使うには事前にインストールが必要です。

Carthageは次のサイトから入手することができます。 https://github.com/Carthage/Carthage/releases



ダウンロードしたいバージョンのCarthage.pkgをダウンロードします。ここでは0.26.2を使います。

ダウンロードしたpkgファイルをダブルクリックで開きインストールします。 インストーラーに従って進めます。通常のアプリと一緒で特に変わった手順は必要ありません。

	🤿 のインストール		
	ようこそインストーラへ		
<ul> <li>はじめに</li> <li>インストール先</li> <li>インストールの種類</li> <li>インストール</li> </ul>	このソフトウェアをインストールするために必要な す。	≵手順が表示されま	
● 概要			
	戻る	続ける	──「続ける」をクリッ
L			
	<b>愛</b> のインストール		







# 

「開発元が未確認」という警告が表示される場合は、セキュリティ設定から「このまま開く」を選択するこ とでインストールできます。

~~~~	"Carthage.pkg"は、開発元が未確認のため開けま せん。	-
	"セキュリティ"環境設定でインストールが許可されているの は、App Storeと確認済みの開発元からのアプリケーションの みです。	
	このファイルは"Google Chrome.app"により今日の14:10に <b>github.com</b> からダウンロードされました。	
?	ОК	「OK」をクリックして警告画面を閉じ

•••		)	システム	環境設定	(	Q 検索	
File New 〇마	デスクトップと スクリーンセーバ	oggo Dock	Mission Control	言語と地域	セキュリティ と プライバシー	Q Spotlight	通知
<u></u> ディスプレイ	省エネルギー	キーポード	。 マウス	トラックパッド	プリンタと スキャナ	サウンド	起動ディスク
iCloud	@ インターネット アカウント	App Store	ネットワーク	Bluetooth	機能拡張	(秋) 共有	
ユーザと グループ	ペアレンタル コントロール	Siri	日付と時刻	Time Machine	アクセシ ビリティ		
چي Java							





#### Note

2018年3月時点では Carthageが悪意のあるプログラムでないことを確認しています。ただし、この設定は セキュリティを一時的に弱めることになるため、開くプログラムが間違っていないかなどを確認し、慎重に行 なってください。

# 

依存管理ツールはコマンドラインから使用します。苦手意識がある方も、必要なコマンドは数コマンドなの で、本書を見ながら実行してみてください。

アプリケーションフォルダのユーティリティの中にあるターミナル.appを起動します。

	tobioka — -bash — 80×24				
Last login: Fri Oct 13 21:35:08 on ttys001 Tatsuyas-MBP:~ tobioka\$ ▋					

以下のコマンドを入力してエンターキーを押します。先頭の\$記号はターミナルでの入力を表す記号で実際 に入力するのはその後に続く内容のみです。√ジはreturn (Enter)キーを示すものです。

ターミナル		
\$ carthage version $↔$		

# 実行結果

0.26.2

先程ダウンロードしたバージョンが表示されていればインストールは成功です。

Tobloka — -bash — 80×24
Last login: Fri Oct 13 21:35:08 on ttys001 Tatsuyas-MBP:∼ tobloka\$ carthage version 0.26.2 Tatsuyas-MBP:∼ tobloka\$ ■

# ライブラリの管理方法 ………………………………………………………………

Carthageによるライブラリ管理は以下の流れで行います。少し複雑ですが、ライブラリ追加のたびに毎回行なう作業なのでいずれ慣れるでしょう。

- 1. Cartfileの作成
- 2. ビルド
- 3. Linked Frameworks and Librariesの追加
- 4. Run Scriptの設定

5. Input Filesの設定

6. Output Filesの設定

7. ライブラリを使用する

# サンプルファイル ………

種類	ファイル
プロジェクト	Bonus/Chapter4/AlamofireExample.xcodeproj
Storyboard	変更しません
ソース	Bonus/Chapter4/AlamofireExample/AlamofireExample/ViewController.swift

# 

Cartfileというファイルに使用したいライブラリを記載します。このファイルはプロジェクトフォルダー (プロジェクト名.xcodeprojファイルがあるフォルダー)に作ります。



iOS watchOS tvO	S macOS			Filter
Resource				
Stringsdict File				
Other				
	S	7" CONFIG		h
Empty	Assembly File	Configuration Settings File	Markdown File	PCH File
T				
SHELL				
Shell Script				
Cancel			Previo	ous Next

	Save As: Cartfile		
	AlamofireExample	Q Search	
Favorites Carlo Cloud Drive Cloud Drive Carlo Cl	AlamofireExample		「.xcodeproj」ファイルがあるフォルダーに 保存します。
	Group AlamofireExample		
New Folder		Cancel Create	

例えば、Alamofire (<u>https://github.com/Alamofire/Alamofire</u>) というライブラリのバージョン 4.5.1 を使いたい場合は以下のように記載します。

サンプルプログラム Bonus/Chapter4/AlamofireExample/Cartfile

github "Alamofire/Alamofire" == 4.5.1

Chapter 4 外部ライブラリを使う

複数行書くことで複数のライブラリを使用できます。

#### Note

Alamofireは最も有名な Swift製のネットワーク通信ライブラリです。Chapter 20では URLSessionを用いた通信方法を紹介しましたが、複雑な通信をする際はAlamofireの利用を検討してみてもよいでしょう。

2. ビルド ………

以下のコマンドを叩いてライブラリをダウンロード及びビルドします。

cdはチェンジディレクトリーの意味でプロジェクトフォルダーまで移動するコマンドです。cd(c、d、半 角スペース)まで入力した後、ターミナルにフォルダーをドラッグ&ドロップすれば場所が入力されるので、 そのままreturnキーを押せばOKです。

(/Users/ユーザー名/フォルダー名/プロジェクト名のような形式になります。)
# ① [Cartfile]を右クリックします。 Cartfile Show in Finder ● Alamofi ● Appt ● View ● Main Show File Inspector





ター	-ミナル	
\$	cd プロジェクトフォルダーの場所✔	
\$	carthage updateplatform	iOS↩

実行結果
*** Fetching Alamofire
*** Checking out Alamofire at "4.5.1"
<pre>*** xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch</pre>
hv6wr_h7d4h0000gp/T/carthage-xcodebuild.2B4LYo.log
*** Building scheme "Alamofire iOS" in Alamofire.xcworkspace

errorなどのメッセージが出ておらず、Cartfile.resolvedというファイルと Carthageというフォ ルダーが作られていれば成功しています。

• • •		AlamofireExample	
$\langle \rangle$		<b>☆</b> ▲ □	Q. 検索
よく使う用 (m) AirDrop ローマイファイル へ) ICloud Drive ヘ、アブリケーション ローズクトップ デスクトップ の クサクンロード デバイス (の) リモートディスク タグ Red Orange Yellow	AlamofireExample	AlamofreExample     AlamofreExae.xcodoproj     Cartrile     Cartrile     Carthage	
Groop			

**Carthage**フォルダーの中をBuild、iOSとたどるとライブラリ名.frameworkというファイルが見つかります。今回であればAlamofire.frameworkです。

< >         SS ≡ □□□1 □ □ □ · ◆ · ①         Q. ℓ/π           >         >         SS ⇒ □ □ □ □ □ □ · · · · · · · · · · · · ·	
Build P iOS P 5C55D430c	
	csymbolmap nework. Nework.dSYM bcsymbolmap

そのAlamofire.frameworkをXcodeの「Linked Frameworks and Libraries」にドラッグ&ドロッ プして追加します。

General		
E Ocheran	Capabilities Resource Tags Info Build Settings Build Phases Build	Id Rules
PROJECT		
AlamofireExample	▼ Deployment Info	
TARGETS	Deployment Target 11.0	
$p\Delta_{\underline{r}}$ AlamofireExample	Devices Universal	
	Main Interface Main	
	Device Orientation 🖾 Portrait	
	Upside Down	
	Landscape Left	
	Landscape Right	
	Status Bar Style Default	
	Hide status bar	
	Requires full screen	
	▼ App Icons and Launch Images	
	App Icons Source AppIcon	
	Jaunch Images Source Lise Asset Catalog	
	caunci integra bonce lose vaser carang	
	Launch Screen File LaunchScreen	
	▼ Embedded Binaries	
	Add embedded binaries here	
	+ -	
	▼ Linked Frameworks and Libraries	
	Name Status	
	Alamofire.framework Regulted 0	最下部の「Linked Frameworks and Libraries」に
		┛────────────────────────────────────
+ - ® Ellter	+ -	

「Build Phases」を開き、左上の「+」から「New Run Script Phase」を選択します。

		①[Build F	hases」をく	<b>フリックし</b>	ます。		
Genera	Capabilities Resource Tags	Info Build Set	tings Build Phases	Build Rules			
PROJECT	Ð		Silter			― ②「+」をクリックします。	
AlamofireExample	New Copy Files Phase						
AlamatizaExample	New Run Script Phase						
but Manonieczampie	New Headers Phase New Copy Bundle Resources Pha	ise			×	–∣③∣New Run Script Phase」 ৰ	:選択します
	New Compile Sources Phase				×		
	New Link Binary With Libraries Ph New Build Carbon Resources Pha	hase					
	Contraction and a solution of the				×		
			S Filter				
- Target Dependence	ies (0 items)		🕞 Filter				
Target Dependence Compile Sources	:ies (0 items) ;2 items)		Pilter		×		
Target Dependence  Compile Sources  Link Binary With L	:les (O items) (2 items) ibraries (1 item)		S Filter		× × ×		
Target Dependence Target Dependence Compile Sources Link Binary With L Copy Bundle Resc	cles (0 items) (2 items) Ibraries (1 item) urces (4 items)		🕞 Filter		× × ×		
Target Dependent  Compile Sources  Link Binary With L  Copy Bundle Resc	iles (D items) (2 items) ibraries (1 item) urces (4 items)		( Filter		× × ×	「Durp Soviet」が泊加されます	]

追加された「Run Script」の左側の三角形をクリックし、「Shell」の右下のテキストボックスに以下の内容を入力します。

# /usr/local/bin/carthage copy-frameworks

Run Script		×		たクロッ	17	問きせす	٦
	Shell //bin/sh			2790	////	ннсь ус	2
	1 Type a script or drag a script file from your workspace to insert its path						
	Show environment variables in build log						
	Input Files						
	Add input files here						
	+ -						
	Output Files						
	Add output files here						
	+ -						
Run Script		×					
7 Run Script	Shell /bin/sh	×					
<sup>7</sup> Run Script	Shell /bin/sh	×	- 上記:	スクリフ	プトの内	容を入力	
<sup>7</sup> Run Script	Shell /bin/sh //usr/local/bin/carthage copy-frameworks	×	- 上記:	スクリフ	プトの内	容を入力	しまう
Run Script	Shell /bin/sh /usr/local/bin/carthage copy-frameworks Show environment variables in build log Run script only when installing Input Files	×	—上記:	スクリフ	プトの内	容を入力	しま <sup>-</sup>
Run Script	Shell /bin/sh //usr/local/bin/carthage copy-frameworks Show environment variables in build log Now script only when installing Input Files Add input files here	x	—上記:	スクリフ	プトの内	容を入力	しまっ
' Run Script	Shell       /bin/sh         /usr/local/bin/carthage       copy-frameworks         Show environment variables in build log	×	—上記.	スクリフ	プトの内	容を入力	します
' Run Script	Shell       /bin/sh         /usr/local/bin/carthage copy-frameworks         Show environment variables in build log         Run script only when installing         Input Files         Add input files here         +         Output Files	×	—上記:	スクリフ	プトの内	容を入力	しまう
' Run Script	Shell //bin/sh //usr/local/bin/carthage copy-frameworks Show environment variables in build log Nun script only when Installing Input Files Add Input files here + Output Files Add output files here	x	—上記;	スクリフ	プトの内	容を入力	します

# 

スクリプトの下の「Input Files」の「+」をクリックし、以下の内容を入力します。

# \$(SRCROOT)/Carthage/Build/iOS/Alamofire.framework

Run Script	×	
	Shell         /bin/sh           1         /usr/local/bin/carthage copy-frameworks	
	Show environment variables in build log Run script only when installing Input Files	
	S(SRCROOT)/Carthage/Build/IOS/Alamofire.framework	②上記の内容を入力します。
	Output Files	
	Add output files here	
	+ -	

# 6. Output Filesの設定 ……

「Output Files」にも同様にして、以下の内容を追加します。

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/Alamofire.framework

▼ Run Script	x	
	Shell         //bin/sh           1         /usr/local/bin/carthage copy-frameworks	
	Show environment variables in build log           Run script only when installing           Input Files           \$(SRCROOT)/Carthage/Build//OS/Alamofire.framework	
	Output Files  S(BUILT_PRODUCTS_DIR)/S(FRAMEWORKS_FOLDER_PATH)/Alamofire.framework	②上記の内容を入力します。
	+ -	

# 7. ライブラリを使用する …………………………………………………………

これで準備は整いました。Alamofireはネットワークのライブラリなので、例として http://www.socym.co.jp/のコンテンツを取得してみましょう。

ライブラリをimportします。

Chapter 4 外部ライブラリを使う

サンプルプログラム Bonus/Chapter4/AlamofireExample/AlamofireExample/ViewController.swift

import Alamofire

viewDidLoadで通信します。

```
サンプルプログラム Bonus/Chapter4/AlamofireExample/AlamofireExample/ViewController.swift

override func viewDidLoad() {

    super.viewDidLoad()

    Alamofire.request("http://www.socym.co.jp/").responseString {

    response in self.textView.text = response.result.value

    }

}
```

HTTPS通信ではないのでATSで許可します。



このような手順で外部ライブラリを利用したアプリを開発することができます。他のライブラリでも基本的 に手順は同じです。

本 PDFの Section 3.3以降では、定番のライブラリをいくつか紹介します。よく使われているライブラ リは品質が担保され、安心感があります。選定の参考にしてください。Objective-Cで開発されたライブラ リもたくさんありますが、本書ではSwiftで開発されたライブラリを紹介します。

### Note

利用するライブラリを増やし過ぎると、プロジェクトの維持・管理が煩雑になりますので注意してください。



# ・ サンプルアプリ ……………………………………



# 

種類	ファイル
プロジェクト	Bonus/Chapter4/PKHUDExample.xcodeproj
Storyboard	変更しません
ソース	Bonus/Chapter4/PKHUDExample/PKHUDExample/ViewController.swift

# PKHUDについて

通信中や時間のかかるデータ処理など、やむを得ずアプリのユーザーを待たせるケースがあります。その際 にはローディング表示をしてあげると親切です。

単純なインジケーターの表示なら、UIActivityIndicatorViewを使って表示することもできます。しかし、 ローディング表示を最前面に出したり、画面をロック(操作不可に)したりするのは、意外に手間がかかりま す。

**PKHUD**(<u>https://github.com/pkluz/PKHUD</u>)を使うとローディング表示を簡単に実現することが できます。

サンプルアプリの作成 ………………………………………………………

## 1. Cartfileの作成

Cartfileに以下を追記します。

サンプルプログラム Bonus/Chapter4/PKHUDExample/Cartfile

github "pkluz/PKHUD" == 5.0.0

## 2. ビルド

updateコマンドを実行し、ビルドします。

### ターミナル

- \$ cd プロジェクトフォルダーの場所→
- \$ carthage update --platform iOS↓

# 実行結果

\*\*\* Fetching PKHUD
\*\*\* Checking out PKHUD at "5.0.0"
\*\*\* Checking out PKHUD at "5.0.0"
\*\*\* xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch
hv6wr\_h7d4h0000gp/T/carthage-xcodebuild.2b0GT7.log
\*\*\* Building scheme "PKHUD" in PKHUD.xcodeproj

### 3. Linked Frameworks and Librariesの追加

**Carthage/Build/iOS/PKHUD.framework**を「Linked Frameworks and Libraries」にドラッグ& ドロップします。

### 4. Run Script

Run ScriptもAlamofireの時と同様に追加します。

# スクリプト

/usr/local/bin/carthage copy-frameworks

### Input Files

\$(SRCROOT)/Carthage/Build/iOS/PKHUD.framework

### **Output Files**

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/PKHUD.framework

#### 5. HUDの表示

準備が整ったのでPKHUDを使用してHUDを表示してみます。ライブラリをimportします。

サンプルプログラム Bonus/Chapter4/PKHUDExample/PKHUDExample/ViewController.swift

import PKHUD

viewDidAppearで表示します。ローディング表示し3秒後に消した後、成功の表示を1秒間出しています。

#### サンプルプログラム Bonus/Chapter4/PKHUDExample/PKHUDExample/ViewController.swift

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)
    HUD.show(.progress) // 読み込み中の表示
    HUD.hide(afterDelay: 3.0) { _ in // 3秒後に今の表示を消す
        HUD.flash(.success, delay: 1.0) // 成功を表示して1秒後に消す
    }
}
```





# 

種類	ファイル
プロジェクト	Bonus/Chapter4/EurekaExample.xcodeproj
Storyboard	変更しません
ソース	Bonus/Chapter4/EurekaExample/EurekaExample/ViewController.swift

ユーザーに何か入力させたい場合にはフォームが必要です。HTMLではとても簡単なこの作業ですが、 iOSアプリでは少し大変です。

タイトルと本文を入力するフォームを考えてみましょう。

フォームらしい見た目にするには TableViewを使う必要があります。そして各セルに TextFieldや TextViewを配置して、Delegateなどを適切に処理する必要があります。これはとても骨の折れる作業です。 フォーム機能がアプリのメイン機能ではない場合は特にライブラリに頼るのが得策です。

**Eureka** (<u>https://github.com/xmartlabs/eureka</u>) を使えば iOS標準アプリのようなフォームが簡 単に作れます。

1サンプルファイルの作成 ………………………………………………………

### 1. Cartfileの作成

Cartfileに次の内容を記載します。

#### サンプルプログラム Bonus/Chapter4/EurekaExample/Cartfile

github "xmartlabs/Eureka" == 4.0.1

## 2. ビルド

次のようにビルドします。

## ターミナル

- \$ cd プロジェクトフォルダーの場所→
- \$ carthage update --platform iOS↓

# <mark>実行結果</mark> \*\*\* Cloning Eureka \*\*\* Checking out Eureka at "4.0.1" \*\*\* xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch hv6wr\_h7d4h0000gp/T/carthage-xcodebuild.m7VFbn.log \*\*\* Building scheme "Eureka" in Eureka.xcworkspace

## 3. Linked Frameworks and Librariesの追加

**Carthage/Build/iOS/Eureka.framework**を Linked Frameworks and Librariesにドラッグ & ドロップします。

# 4. Run Script

次のように設定します。

# スクリプト

/usr/local/bin/carthage copy-frameworks

### Input Files

\$(SRCROOT)/Carthage/Build/iOS/Eureka.framework

### **Output Files**

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/Eureka.framework

## 5. フォームの表示

タイトルと本文が入力できるフォームを表示します。 Eurekaを import します。

サンプルプログラム Bonus/Chapter4/EurekaExample/EurekaExample/ViewController.swift

import Eureka

FormViewControllerを継承します。

サンプルプログラム Bonus/Chapter4/EurekaExample/EurekaExample/ViewController.swift

class ViewController: FormViewController {

フォームの設定を行います。

```
サンプルプログラム Bonus/Chapter4/EurekaExample/EurekaExample/ViewController.swift
```

```
override func viewDidLoad() {
    super.viewDidLoad()
    // タイトルを入力する Row (1行)
    let titleRow = TextRow { row in
        row.title = "タイトル"
       row.placeholder = "タイトルを入力"
    }
    // 本文を入力するRow(複数行)
    let bodyRow = TextAreaRow { row in
        row.placeholder = "本文を入力"
    }
    // 入力用のセクションを作ってRowを追加
    let section = Section()
    section.append(contentsOf: [
       titleRow.
        bodyRow
    ])
    form.append(contents0f: [
       section
    ])
}
```

ナビゲーションバーの「Save」をクリックすると内容をアラートで表示します。

#### サンプルプログラム Bonus/Chapter4/EurekaExample/EurekaExample/ViewController.swift

```
@IBAction func onTap(_ sender: Any) {
    // タイトルを取得
    let title = (form.allRows[0] as? TextRow)?.value ?? ""
    // 本文を取得
    let body = (form.allRows[1] as? TextAreaRow)?.value ?? ""
    let message = """
        title: \(title)
        body: \(body)
        """
    let alertController = UIAlertController(
```

```
このような数行のプログラムを書くことでフォームを表示させることができます。
```

Swiftは独自の演算子を定義できます。Eurekaではそれが活用されており先ほどと同じフォームを以下のようなプログラムでも書くことができます。

```
サンプルプログラム Bonus/Chapter4/EurekaCustomOperator/EurekaCustomOperator/ViewController.swift
```

```
let titleRow = TextRow { row in
    row.title = "タイトル"
    row.placeholder = "タイトルを入力"
}
let bodyRow = TextAreaRow { row in
    row.placeholder = "本文を入力"
}
let section = Section()
section
    <<< titleRow
    <<< bodyRow</pre>
```

どちらが読みやすいかは好みによるでしょう。

筆者は独自演算子を学習するコストを考慮して、基本的には使わない方針にしています。



サンプルアプリ	
Carrier 🗢 12:40 PM 👘 🕈	1
	画像を使わずにアイコン付きの各種ボタンを表示します。

種類	ファイル
プロジェクト	Bonus/Chapter4/SwiftlconFontExample.xcodeproj
Storyboard	Bonus/Chapter4/SwiftIconFontExample/SwiftIconFontExample/Base.lproj/ Main.storyboard
ソース	Bonus/Chapter4/SwiftIconFontExample/SwiftIconFontExample/ ViewController.swift

アプリのボタンなどにアイコンを使いたいことがあります。 標準でいくつかのアイコン素材が提供されていますが、充分ではないこともあります。

画像のアイコン素材を使うと、色やサイズの変更などに手間がかかります。そんな時に便利なのがアイコン フォントです。文字としてアイコンを表示できるので、色やサイズなどを柔軟に指定することができます。

ここでは SwiftIconFont (<u>https://github.com/0x73/SwiftIconFont</u>) でアイコンを表示させてみ ます。

# 

# 1. Cartfileの作成

Cartfileに次の内容を記載します。

サンプルプログラム Bonus/Chapter4/SwiftIconFontExample/Cartfile

github "0x73/SwiftIconFont" == 2.7.2

# 2. ビルド

次のようにビルドします。

# ターミナル

\$ cd プロジェクトフォルダーの場所→

\$ carthage update --platform iOS↓

## 実行結果

\*\*\* Cloning SwiftIconFont
\*\*\* Checking out SwiftIconFont at "2.7.2"
\*\*\* xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch
hv6wr\_h7d4h0000gp/T/carthage-xcodebuild.hM2LCS.log
\*\*\* Building scheme "SwiftIconFont" in SwiftIconFont.xcodeproj

### 3. Linked Frameworks and Librariesの追加

**Carthage/Build/iOS/SwiftIconFont.frameworkをLinked Frameworks and Libraries**に ドラッグ&ドロップします。

### 4. Run Script

次のように設定します。

# スクリプト

/usr/local/bin/carthage copy-frameworks

#### Input Files

\$(SRCROOT)/Carthage/Build/iOS/SwiftIconFont.framework

#### Output Files

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/SwiftIconFont.framework

### 5. アイコンの表示

SwiftlconFontをimportします。

サンプルプログラム Bonus/Chapter4/SwiftIconFontExample/SwiftIconFontExample/ViewController.swift

import SwiftIconFont

viewDidLoadでアイコンを設定したLabelをViewに追加します。ナビゲーションバーやツールバーにも 表示します。

#### サンプルプログラム Bonus/Chapter4/SwiftIconFontExample/SwiftIconFontExample/ViewController.swift

```
override func viewDidLoad() {
    super.viewDidLoad()
    // スマートフォンのアイコンを持つラベル
    let label = UILabel(
        frame: CGRect(x: 20.0, y: 80.0, width: 50.0, height: 50.0)
    )
    label.font = UIFont.icon(from: .FontAwesome, ofSize: 50.0)
```

```
label.text = String.fontAwesomeIcon("mobile")
view.addSubview(label)

// ユーザーのアイコンを持つナビゲーションバーアイテム
let navItem = UIBarButtonItem()
navItem.icon(from: .FontAwesome, code: "user", ofSize: 20.0)
navigationItem.rightBarButtonItem = navItem

// カレンダーのアイコンを持つツールバーアイテム
let toolItem = UIBarButtonItem()
toolItem.icon(from: .FontAwesome, code: "calendar", ofSize: 20.0)
toolbarItems = [toolItem]
```

このように画像を使わずアイコンを簡単に表示することができます。

#### Note

}

ここでは FontAwesomeというフォントを使いました。次のページで FontAwesomeフォントで利用できるアイコンの一覧を確認することができます。





#### \_\_\_\_\_ サンプルアプリ …………

パスワードなど       保存         別除       読込         パスワードなど       ハカレたテキストを保存・読込・削除します。	Carrier 🗢	1:05 PM	· *	
保存           削除           読込   パスワードなど	パスワードなど	-		
削除		保存		
厳込 パスワードなど		削除	H-	一入力したテキストを保存・読込・削除します。
		読込		
	パスワードな	Ľ		

# 

種類	ファイル
プロジェクト	Bonus/Chapter4/KeychainAccessExample.xcodeproj
Storyboard	Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/Base. lproj/Main.storyboard
ソース	Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ ViewController.swift

書籍版の Chapter 18 で紹介した UserDefaults やファイルにそのまま保存する方法では情報が暗号化されません。そのため機密情報、例えばパスワードなどの認証情報をその方法で保存するのは安全とはいえません。

iOSにはキーチェーンというセキュアな情報保存の仕組みが用意されているのですが、とても複雑でこれ を自分で操作するのは大変です。

KeychainAccess (<u>https://github.com/kishikawakatsumi/KeychainAccess</u>) というライブラ リを使うと簡単にキーチェーンに情報を保存できます。

### 1. Cartfileの作成

次の内容をCartfileに記載します。

サンプルプログラム Bonus/Chapter4/KeychainAccessExample/Cartfile

github "kishikawakatsumi/KeychainAccess" == 3.1.0

## 2. ビルド

次のようにビルドします。

# ターミナル

- \$ cd プロジェクトフォルダーの場所→
- \$ carthage update --platform iOS↓

## 実行結果

\*\*\* Cloning KeychainAccess
\*\*\* Checking out KeychainAccess at "v3.1.0"
\*\*\* xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch
hv6wr\_h7d4h0000gp/T/carthage-xcodebuild.cF1gkv.log
\*\*\* Building scheme "KeychainAccess" in KeychainAccess.xcworkspace

### 3. Linked Frameworks and Librariesの追加

Carthage/Build/iOS/KeychainAccess.frameworkを Linked Frameworks and Libraries にドラッグ&ドロップします。

### 4. Run Script

次のように設定します。

# スクリプト

/usr/local/bin/carthage copy-frameworks

### Input Files

\$(SRCROOT)/Carthage/Build/iOS/KeychainAccess.framework

## Output Files

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/KeychainAccess.framework

#### 5. 機密データの保存

キーチェーンに情報を保存してすぐ読み出すだけの単純な処理を行います。 KeychainAccessをimportします。

サンプルプログラム Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ViewController.swift

import KeychainAccess

viewDidLoadでパスワードを保存します。

```
サンプルプログラム Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ViewController.swift
```

```
// 複数のメソッドで使うのでプロパティとしてインスタンスを作成
let keychain = Keychain()
@IBAction func onTapSave(_ sender: Any) {
    guard let text = textField.text, !text.isEmpty else { return }
    keychain["secret"] = text // "secret"をキーとしてテキストを保存
}
```

viewDidAppearで先程保存したパスワードを読み込んでラベルに表示します。

サンプルプログラム Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ViewController.swift
@IBAction func onTapLoad(\_ sender: Any) {
 // "secret"をキーとして保存された情報を取得
 label.text = keychain["secret"]
}

nilを代入すれば削除できます。

また、以下のようにremoveメソッドを使っても、保存した情報を削除することができます。

サンプルプログラム Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ViewController.swift keychain.remove("secret")

サンプルプログラム Bonus/Chapter4/KeychainAccessExample/KeychainAccessExample/ViewController.swift

@IBAction func onTapRemove(\_ sender: Any) {
 keychain["secret"] = nil // "secret"をキーと保存された情報を削除
}

UserDefaultsと変わらない手軽さですが、これで情報を安全に保存することができます。

#### Note

KeychainAccess ライブラリは、この他にも iCloud による同期や、Touch ID による認証などもサポートしています。



<u>サンプ</u>ルアプリ ……………

Carrier 🗢	1:45 PM	• •
スティーブ		
ジョニー		

# サンプルファイル ………

種類	ファイル
プロジェクト	Bonus/Chapter4/RealmExample.xcodeproj
Storyboard	Bonus/Chapter4/RealmExample/RealmExample/Base.lproj/Main.storyboard
ソース	Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift

アプリが扱うデータの量が多くなってくると、UserDefaultsやファイルへの保存だけでは管理が大変に なってきます。また検索などの機能も用意されていないため、これらの方法はそもそも大量のデータを管理す るには向いていません。そこでデータベースを使うことになります。

iOSには Core Dataという標準のデータベースがありますが、利用する手順が複雑です。ここでは簡単に 使える Realm (<u>https://github.com/realm/realm-cocoa</u>) を紹介します。Realmには Androidなど iOS以外のプラットフォームでも動くという特徴があります。

# サンプルアプリの作成 ………

### 1. Cartfileの作成

次の内容をCartfileに記載します。

サンプルプログラム Bonus/Chapter4/RealmExample/Cartfile

github "realm/realm-cocoa" == 3.0.1

## 2. ビルド

次のようにビルドします。ここまではこれまでと同様の手順です。

# ターミナル

\$ cd プロジェクトフォルダーの場所→

\$ carthage update --platform iOS↓

### 実行結果

\*\*\* Fetching realm-cocoa

\*\*\* Downloading realm-cocoa.framework binary at "v3.0.1"

\*\*\* xcodebuild output can be found in /var/folders/xx/2bhj86b17j9ch hv6wr\_h7d4h0000gp/T/carthage-xcodebuild.6xR4B2.log

### 3. Linked Frameworks and Librariesの追加

Carthage/Build/iOS/RealmSwift.frameworkとCarthage/Build/iOS/Realm.frameworkを Linked Frameworks and Librariesにドラッグ&ドロップします。

2つのフレームワークが必要なので注意してください。

### 4. Run Script

以下のように設定します。ここも2つのフレームワークを指定する必要があります。

### スクリプト

/usr/local/bin/carthage copy-frameworks

### Input Files

\$(SRCROOT)/Carthage/Build/iOS/Realm.framework
\$(SRCROOT)/Carthage/Build/iOS/RealmSwift.framework

### Output Files

\$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/Realm.framework \$(BUILT\_PRODUCTS\_DIR)/\$(FRAMEWORKS\_FOLDER\_PATH)/RealmSwift.framework

#### 5. データベースの利用

ユーザー情報を2人分保存してそれを一覧表示するサンプルです。 importします。インポートは1つで大丈夫です。

サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift

import RealmSwift

テーブルビューコントローラーを継承します。Main.storyboard でも元々配置されている View Controllerの代わりに Table View Controllerを使います。詳しい手順は書籍版の Chapter 13を参照してください。

サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift

class ViewController: UITableViewController {

データーベースから取得した結果を保存する変数を用意します。

サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift

```
// データベースから取得したPerson一覧を保持するプロパティ
var people: Results<Person>!
```

viewDidLoadではデータベースからユーザーを全件取得し、結果が空なら2件表示する処理を行っています。この処理のあとTableViewを更新するためにreloadDataを呼んでいます。

```
サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift
override func viewDidLoad() {
    super.viewDidLoad()
    let realm = try! Realm()
                                       // Realmのインスタンスを作成
    people = realm.objects(Person.self) // Person一覧を取得
                                       // Person一覧が空なら
    if people.isEmpty {
        let person1 = Person()
        person1.name = "スティーブ"
        let person2 = Person()
        person2.name = "ジョニー"
        try! realm.write {
                                       // 2人のPersonを追加
            realm.add(person1)
            realm.add(person2)
        }
    }
    tableView.reloadData()
                                       // TableViewを再読込
}
```

TableViewに表示するためのDataSource、Delegateの設定も必要です。

サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/ViewController.swift	
<pre>override func tableView(_ tableView: UITableView,</pre>	
<pre>numberOfRowsInSection section: Int) -&gt; Int {</pre>	
return people.count // 一覧の数だけセルを表示	
}	

```
override func tableView(_ tableView: UITableView,
cellForRowAt indexPath: IndexPath) -> UITableViewCell {
let cell = tableView.dequeueReusableCell(
withIdentifier: "reuseIdentifier", for: indexPath
)
// 名前をセルに表示
let person = people[indexPath.row]
cell.textLabel?.text = person.name
return cell
}
```

保存するデータはこのように定義します。

サンプルプログラム Bonus/Chapter4/RealmExample/RealmExample/Person.swift

import Foundation
import RealmSwift
class Person: Object {
 @objc dynamic var name = ""
}

単純な例ですが、データベースへの保存、データベースからの読み込みと表示ができました。

Realmに保存するデータは、Objectクラスを継承する必要があります。

また、プロパティには「@objc dynamic」を付けるのを忘れないようにしましょう。これによって Realmがプロパティを管理できるようになります。

#### Note

Realmは、検索などデータベースに必要な様々な機能を提供しています。 少し情報は遅くなりますが、日本語ドキュメントも用意されていますので、活用を検討してみてください。 https://realm.io/jp/docs/swift/latest/



# サンプルアプリ ……………

Carrier	2:04 PM	<b>*</b>
	Settings	
3.4	oonorai	· · ·
	Privacy	>
0	Maps	>
	Safari	>
N	News	>
$\wedge$	Siri	>
٠	Photos	>
5	Game Center	>
$\checkmark$	Developer	>
	LicensePlistExample	>



Carrier 🗢	2:04 PM	· *	Carri
🗸 Back	Licenses		< L
Alamofire		>	Cop (htt
Eureka		>	Per
KeychainAc	cess	>	of t "So in t
PKHUD		>	limi to u
realm-cocoa	3	>	cop the
SwiftIconFo	nt	>	The
			all o
			TH
			IMF WA
			FIT NO
			AU AN
			LIA
			OU
			THI

Carrier 🗢	2:04 PM
🕻 Licenses	Alamofire
Copyright (c) 2014 (http://alamofire.o	1-2017 Alamofire Software Foundation rg/)
Permission is here	by granted, free of charge, to any
of this software ar "Software"), to de	i copy id associated documentation files (the al
in the Software wi	thout restriction, including without
to use, copy, mod	, fy, merge, publish, distribute,
copies of the Soft	ware, and to permit persons to whom
the Software is furnished to do so	, subject to the following conditions:
The above copyrig	ht notice and this permission notice
all copies or subst	antial portions of the Software.
THE SOFTWARE I	S PROVIDED "AS IS", WITHOUT
IMPLIED, INCLUDI	VY KIND, EXPRESS OR NG BUT NOT LIMITED TO THE
WARRANTIES OF	MERCHANTABILITY,
NONINFRINGEME	NT. IN NO EVENT SHALL THE
AUTHORS OR CO	PYRIGHT HOLDERS BE LIABLE FOR
LIABILITY, WHETH	IER IN AN ACTION OF CONTRACT,
TORT OR OTHERW	VISE, ARISING FROM,
THE USE OR OTH	ER DEALINGS IN
THE SOFTWARE.	

iOSの「設定」アプリにライセンス情報を表示させます。

種類	ファイル
プロジェクト	Bonus/Chapter4/RealmExample.xcodeproj
Storyboard	変更しません
ソース	変更しません

# LicensePlistについて ………

ここで紹介したライブラリは、すべてオープンソースライセンスに基づき公開されています。とはいえ、無 条件で使えるわけではありません。各ライブラリのライセンスに従う必要があります。

ここで使用したライブラリとそれらのライセンスは次のとおりです。

ライブラリ	ライセンス		
Alamofire	MIT		
PKHUD	MIT		
Eureka	MIT		
SwiftlconFont	MIT		
KeychainAccess	MIT		
Realm	Apache 2.0		

いずれも比較的、制約の少ないライセンスです。使用しているライブラリとそのライセンスを閲覧できるようにしておけば基本的に問題ないでしょう。

LicensePlist (<u>https://github.com/mono0926/LicensePlist</u>) を使えば、iOSの「設定」アプ リにライセンス一覧を簡単に表示させることができます。

## Note

ライブラリをアプリに使用する際には必ずライセンスを確認しましょう。 有志の方々の手によって、以下に各ライセンスの日本語訳が公開されています。

• The MIT License

https://ja.osdn.net/projects/opensource/wiki/licenses%2FMIT\_license

Apache License, Version 2.0 https://ja.osdn.net/projects/opensource/wiki/licenses%2FApache\_License\_2.0

Copyright © 2018 Tatsuya Tobioka / Socym Co., Ltd.

LicensePlistは次のサイトからダウンロードすることができます。 https://github.com/mono0926/LicensePlist/releases

インストール ………



zipが展開されていない場合はダブルクリックして license-plistファイルをダウンロードフォルダに格納 しておきます。(Safariがデフォルト設定の場合は自動で展開されます。)

続いて、次のコマンドを実行します。コマンドを実行するディレクトリーはどこでも構いません。



なし(エラーメッセージが表示されないこと)

iOSの「設定」アプリへ情報を表示させるにはSettings.bundleというファイルが必要です。 次のサイトからダウンロードして展開した Settings.bundleをドラッグ&ドロップしてプロジェクトに 取り込みます。

https://github.com/mono0926/LicensePlist/raw/master/Settings.bundle.zip

Choose options for adding these files:						
Destination: 🗹 Copy if	tems if needed					7
Added folders	groups		_	- [Create gro	oups」を選択します。	
Create	folder references					
Add to targets 🛛 🖂 Lic	ensePlistExample			- 全てチェッ	クONにします。	
				L		
Cancel		Finish				
					1	
			) (?)			
▼ LicensePlistExample	M	Identity and Type				
Cartfile	A	Name Settin	gs.bun	dle		
LicensePlistExample		Location Rela	ive to G			
▼ 🤍 Settings.bundle		Settin	as.bund	dle		
▶ 🛅 en.lproj		Full Path /User	s/tobiok	a/Documents/		
▶ 🛄 ja.lproj		Chan	or24/	Construction of Construction o		
Root.plist	Α	Licen	erz4/ sePlistE	xample/		
AppDelegate.swift	м	Licen	ePlistE	xample/		
ViewController.swift	A	Settin	gs.bund			
Main.storyboard	A	On Demand Resource	Tags			
Assets.xcassets	м	Tags				
LaunchScreen.storyboard	A					
Desdusts	A	Target Membership				
P Products		LicensePlistEx	ample	J		
				-		
このようなファイル構成になり	ノます。					
		File inspecto	or」で	Target Men	nbership」の「Licens	ePlistExample
		にチェックが	2 - 7	ていることを	確認します。	

ライセンス一覧の追加 ………

次のコマンドでライセンス一覧を作成できます。

ターミナル

- \$ cd プロジェクトフォルダーの場所→
- \$ license-plist↓

この Sectionのサンプルプロジェクトに含まれる Carfileには、これまでに紹介した全てのライブラリを 含めてあります。LicensePlistはこれらを基にライセンス情報を生成してくれます。

```
サンプルプログラム Bonus/Chapter4/LicensePlistExample/Cartfile
```

```
github "Alamofire/Alamofire" == 4.5.1
github "pkluz/PKHUD" == 5.0.0
github "xmartlabs/Eureka" == 4.0.1
github "0x73/SwiftIconFont" == 2.7.2
github "kishikawakatsumi/KeychainAccess" == 3.1.0
github "realm/realm-cocoa" == 3.0.0
```

# 実行結果

[2017-10-15T13:58:10.450+09:00] [WARNING] Not found: license plist.yml -- file:///Users/tobioka/Documents/Bonus/Chapter4/LicensePlistExample/. [2017-10-15T13:58:10.455+09:00] [INFO] Start [2017-10-15T13:58:10.456+09:00] [WARNING] not found: Pods/Target%20Supp ort%20Files -- file:///Users/tobioka/Documents/Bonus/Chapter4/LicenseP1 istExample/ [2017-10-15T13:58:10.456+09:00] [INF0] Pods License parse start [2017-10-15T13:58:10.456+09:00] [WARNING] Not found: Pods/Manifest.lock -- file:///Users/tobioka/Documents/Bonus/Chapter4/LicensePlistExample/. [2017-10-15T13:58:10.459+09:00] [WARNING] Not found: Cartfile.resolved -- file:///Users/tobioka/Documents/Bonus/Chapter4/LicensePlistExample/. [2017-10-15T13:58:10.461+09:00] [INFO] Carthage License collect start [2017-10-15T13:58:10.462+09:00] [INFO] Manual License start [2017-10-15T13:58:10.462+09:00] [WARNING] Not found: com.mono0926.Licen sePlist.Output/com.mono0926.LicensePlist.latest\_result.txt -- file:///U sers/tobioka/Documents/Bonus/Chapter4/LicensePlistExample/. [2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: Al amofire, name: Alamofire) [2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: pk luz, name: PKHUD) [2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: xm artlabs, name: Eureka)

<pre>[2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: 0x 73, name: SwiftIconFont)</pre>	
[2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: ki	
shikawakatsumi, name: KeychainAccess)	
[2017-10-15T13:58:10.463+09:00] [INFO] license download start(owner: re	
alm, name: realm-cocoa)	
[2017-10-15T13:58:12.560+09:00] [INFO] Directory created: com.mono0926.	
LicensePlist.Output file:///Users/tobioka/Documents/Bonus/Chapter4/L	
icensePlistExample/	
[2017-10-15T13:58:12.566+09:00] [INFO] End	
[2017-10-15T13:58:12.566+09:00] [INFO]Result	
[2017-10-15T13:58:12.566+09:00] [INFO] # Missing license:	
[2017-10-15T13:58:12.566+09:00] [INFO] None	

作成された com.mono0926.LicensePlist.Output内の com.mono0926.LicensePlist.plistと com.mono0926.LicensePlistを先程のSettings.bundle内にドラッグ&ドロップして追加します。



この状態でアプリを実行します。シミュレータの「設定」アプリを開くと、ライセンス一覧が表示されます。

## Note

en.lproj (英語)、ja.lproj (日本語)のRoot.stringsを編集するとコピーライトの表記などを変更可能です。



これまでのライブラリはアプリの機能を直接改善するものでしたが、最後に番外編として開発を支援する ツールを紹介します。

ある処理を行わせるためのプログラムの書き方は1つとは限りません。そのため、時間をかけて開発したり、 複数の人が協力して開発したりする場合には、プログラムの書き方を統一するルールを決めるのが一般的です。

こうしたルールを意識せずにプログラムを書き進めると、時間の経過とともに一貫性が失われてゆき、次第 に読みづらくなります。

SwiftLint (<u>https://github.com/realm/SwiftLint</u>) というツールを使うと、プログラムの記述スタイ ルを自動的にチェックしてもらえるようになります。読みづらい箇所や、一貫性が失われている箇所などを自 動的に探し出して、指摘してくれるため、結果的に開発速度も向上します。

SwiftLintは次のサイトからダウンロードすることができます。

https://github.com/realm/SwiftLint/releases

•••<	a GitHub, Inc. Č	0 1 0 +	
Latest release	O.23.1: Rewash: Forgotten Load Edition ↓ Ipsim released this 5 days ago - 30 commits to master since this release Breaking • None. Enhancements • None. Bug Fixes • Fix false positive in array_init rule when using a map that doesn't take a closure. Marcelo Fabri #1878 • superfluous_disable_command rule can now be disabled as expected when using // swiftlint:disable_superfluous_disable_command. Marcelo Fabri #1890		
	Downloads		
	T portable_swiftlint.zip	5.03 MB	
	SwiftLint.pkg	5.02 MB	ー   SwiftLint.pkg」をクリックします

インストーラーの指示に従ってインストールします。

	🥪 SwiftLintのインストール	
	ようこそSwiftLintインストーラへ	
● はじめに	このソフトウェアをインストールするために必要な手順が す。	表示されま
● インストール先		
● インストールの種類		
● インストール		
• 概要		
	戻る	続ける

	🥪 SwiftLintのインストール
	インストール先の選択
<ul> <li>はじめに</li> <li>インストール先</li> </ul>	SwiftLintソフトウェアをインストールするディスクを選択してくだ さい。
<ul> <li>インストールの種類</li> <li>インストール</li> <li>概要</li> </ul>	Macintosh HD 2104 GBER
	会社249.78 GB このソフトウェアをインストールするには、15.8 MBのディスク領域が必要です。
	展る「続ける」

	🥪 SwiftLintのインストール	
	"Macintosh HD"に標準インストール	
● はじめに	この操作には、コンピュータ上に15.8 MBの領域が必要です。	
● インストール先	ディスク"Macintosh HD"にこのソフトウェアを標準インストールす	
● インストールの種類	るには、"インストール"をクリックしてください。	
● インストール		
• 概要		
	イジストール光を変更	
	戻る インストール	]+ 「インストール」をクリックしま

Copyright © 2018 Tatsuya Tobioka / Socym Co., Ltd.
<b>インストーラが新しいソフトウェアをインストールしようとしていま す。</b> 許可するにはパスワードを入力してください。	
ユーザ名: Tobioka	
パスワード: ••••••• キャンセル ソフトウェアをインストール	「ソフトウェアをインストール」をクリックし

Section 4.9 ソースコードのスタイルチェック





# セキュリティ設定 ……

Carthageの時と同様、開発元が未確認となってしまうため、インストールを許可する必要があります。





Note

Carthageと同様、セキュリティを弱める操作になるので注意して行なってください。

Run Script .....

次のように設定します。

# スクリプト

```
if which swiftlint >/dev/null; then
   swiftlint
else
   echo "warning: SwiftLint not installed, download from https://github.
com/realm/SwiftLint"
fi
```



Copyright © 2018 Tatsuya Tobioka / Socym Co., Ltd.

Run Script				
·			- 「Run Script」を開きます。	
	Shell /bin/sh			
	1 Type a script or drag a script file from your workspace to insert its path			
	Show environment variables in build log			
	Run script only when installing			
	Input Files			
	Add input files here			
	+ -			
	Output Files			
	Add output files here			
	+ -			
Run Script		×		
	Shell /bin/sh			
	1 if which swiftlint >/dev/null; then	ר		
	<pre>switcint s else echo "warning: SwiftLint not installed, download from https://</pre>	┢	- スクリプトの内容を入力しまし	
	<ul> <li>II</li> <li>Chan and an induction in build land</li> </ul>			
	Run script only when installing			
	Input Files			
	Add input files here			
	+ -			
	Output Files			

設定 ………

+ -

プロジェクトフォルダーに.swiftlint.ymlというファイルを作成します。ここにチェックの設定を書き ます。

🔻 📓 SwiftLintExample	Show in Finder	①右クリックします。
SwiftLintExample	Open with External Editor	
AppDelegate.s	Open As	
ViewController	Show File Inspector	
💽 Main.storyboa		
🔄 Assets.xcasse	New File	────②「New File」を選択します。
LaunchScreen	Add Files to "SwiftLintExample"	

Obassa a tamalata (as	61					
IOS watchOS tvO	s macOS		9	Filter		
Resource						
Stringsdict File						
Other						
		7				
	S	CONFIG		h	1	)「Empty」を選択します。
Empty	Assembly File	Configuration Settings File	Markdown File	PCH File		
T						
Shell Script						
Cancel			Previ	ious Next	2	) [Next]をクリックします。
					1	
	Save As: .swiftlint.	/ml	~			
	Tags:					
		viftLintExample	o Q S	earch	_	
Favorites	AlamofireExample	<ul> <li>SwiftLi</li> <li>SwiftLi</li> </ul>	intExample.xcodeproj		(	D「.xcodeproj」ファイルがあるフォルダ-
<ul> <li>Recents</li> <li>iCloud Drive</li> </ul>	KeychainAccessEx LicensePlistExamp	ample ⊧ e ⊧			2	を選択します。
Applications	PKHUDExample RealmExample	4 14				
Desktop	SwiftlconFontExam SwiftLintExample	ple ►				
Ownloads						
Devices						
	Group D S	vifti intExample				
	Targets 💟 🖂	SwiftLintExample				
New Folder				Cancel Create	C	) Create をクリックします。
The second	Names	that begin v	with a dot "."	' are reserve	ed	
PE	for the	system.				
	If you dec	ide to go ahea th a dot the fil	ad and use a na le will be hidde	ame which		
	begins wi	and dot the fil	ie will be flidde			
			Cancel	Use "."		― [Use "."]をクリックします。

サンプルプログラム Bonus/Chapter4/SwiftLintExample/.swiftlint.yml excluded: - Carthage line\_length: 80 ここではCarthageで入れたライブラリのコードを除外し、1行の文字数を80文字にしています。

アプリをビルド、実行した場合に、プログラムが自動的にチェックされます。

16		let subView = UIView(frame: CG	Rect(x: 0.0, y: 0.0, width: 100.0, height: 100.0))
17		view.addSubview(subView)	A Line Length Violation: Line should be 80 characters or less: currently 88 characters (line length)
18	}		a line longer violation. Line should be be characters of less, carrently be characters (inte_length)

行の長さが長すぎると指摘されています。

ルールの無効化…

SwiftLintの指摘を無効化したい場合は、.swiftlint.ymlに以下のような設定をします。

#### サンプルプログラム swiftlint.yml

disabled\_rules:

- line\_length

この設定によって行が長くても警告が表示されなくなります。たとえば既存のプロジェクトに途中から導入 して、警告が多すぎるためひとまず無視したい、という時に有効でしょう。

また、よくない書き方だとわかっているが、特定の箇所ではやむを得ず指摘を無視したいということもある でしょう。

そのような場合は以下のようなコメントを書くことで無効化できます。

#### サンプルプログラム swiftlint.yml

// swiftlint:disable:next line\_length

このコメントが書かれた次の行は、長さによる警告は行われなくなります。

ここでは便利なライブラリの使い方を紹介しました。これらをぜひ活用して、高機能なアプリを効率的に開 発してみてください。

以上でボーナスPDFは終わりです。詳細は書籍『Swift 4プログラミング入門 iOS 11+Xcode 9対応』 をご購入ください。

Copyright © 2018 Tatsuya Tobioka / Socym Co., Ltd.

150





書名	Swift 4プログラミング入門 iOS 11+Xcode 9対応
著 者	飛岡辰哉
発売開始日	2018年3月19日
発 行 者	ソシム株式会社 <u>http://www.socym.co.jp/</u>
定 価	3,564円(本体3,300円+税)
ISBN	978-4-8026-1153-4
判 型	B5変型判
ページ数	800ページ

# Part 1 iOSアプリ開発の始めかた Chapter 1 Xcode入門

# Part 2 Swiftの文法

Chapter 2	基本的な文法
Chapter 3	関数
Chapter 4	文字列
Chapter 5	配列
Chapter 6	辞書
Chapter 7	セット
Chapter 8	オプショナル
Chapter 9	クラスと構造体
Chapter 10	列挙型

# Part 3 実践iOSアプリ開発

Chapter 11ビューChapter 12主なUIパーツChapter 13ビューの管理Chapter 14オートレイアウトChapter 15図形と画像Chapter 16アニメーションChapter 17ジェスチャーChapter 18データの永続化Chapter 19iOS端末の機能を使うChapter 20Webから情報を取得する





