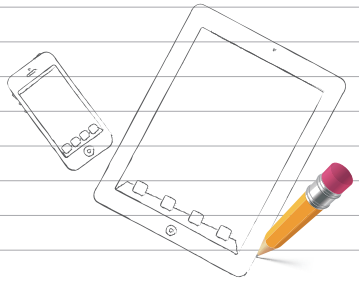
CHAPTER 2

iOSアプリの画面開発の基礎を理解する



この章では、画面の見た目や画面遷移を実装する方法について説明します。
画面と画面遷移とは一体なにかを明確にしてiOSアプリの画面と
画面遷移にどのようなパターンがあるのか、またその作り方の基礎を学びます。
この章で、iOSアプリの画面デザインに欠かすことのできない
Storyboardの使い方を習得することができます。
また、アプリとユーザとのやり取りがどうなっているのか、
そもそもイベントとはいったいなにかということについて学び、
iOSアプリ開発の肝となるイベント駆動プログラミングの理解を深めます。

CHAPTER 2

1 画面開発の基礎知識

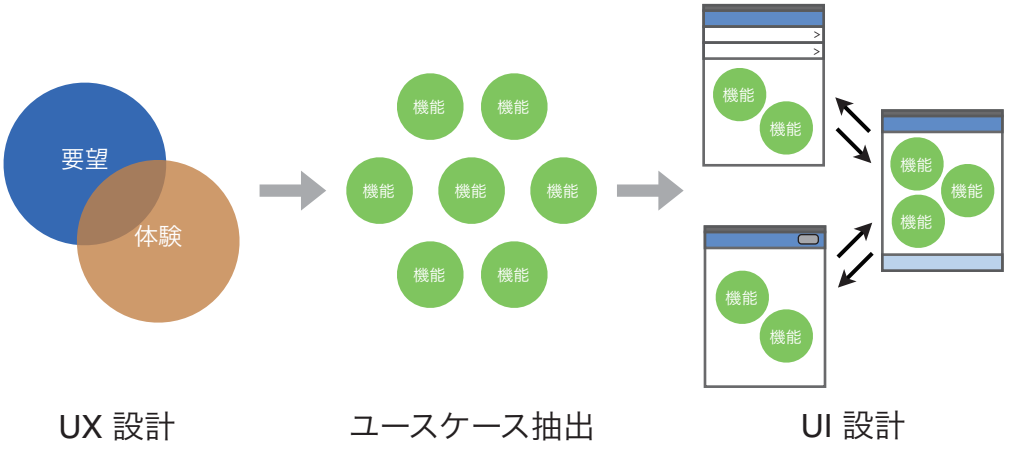
アプリ開発をはじめるにあたってはじめに検討することといえばやはり、アプリの見た目ではないでしょうか。アプリの見た目は、画面にどのようなUI部品を配置して、どのように遷移させるかでほとんどが決まります。UI部品の配置と画面遷移はアプリの使い勝手にも関係しています。同じ機能を持ったアプリであっても見た目と使い勝手でユーザの受ける印象が変わり、アプリの評価に影響します。画面開発はアプリ作成の中で最も重要な部分だといえます。この節では、iOSの標準機能を使ってどのような画面を作ることができるのか、また、実際にどうやって画面を作ればよいのかについて説明します。

2-1-1 ユーザの要望がどのようにアプリに反映されるのか

アプリができる背景には「アプリでこんなことをしたい」というユーザの要望とか、開発者自身の「アプリを通して○○のような素晴らしい体験をユーザに提供したい」という思いが出発点になっています。

ユーザの要望だったり開発者の思いは「UX設計」「ユースケース抽出」「UI設計」という3つの段階に分けてアプリに反映していきます。図にすると以下のような感じです。

▼図2-1-1　UX設計からUI設計までの流れ



「UX設計」「ユースケース抽出」「UI設計」の3つの段階について、詳細を以下に述べます。

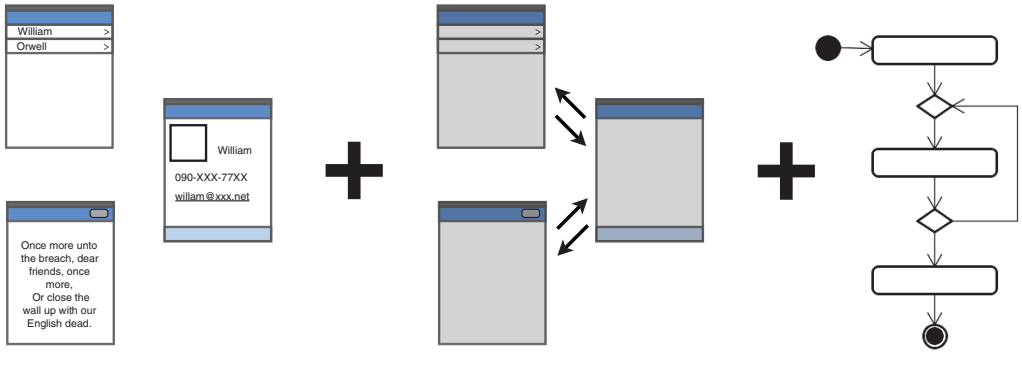
- **UX（ユーザエクスペリエンス）設計**
  ユーザの要望や開発者のこんなもの作りたいという思いを元に、いつ、どこで、誰が使うのか？　アプリが使われる状況を洗い出す。その上でユーザがアプリを使うことによってどんな体験ができるのか、どんな印象を与えるのかをまとめる
- **ユースケース抽出**
  UX設計を元にアプリに必要な機能（ユースケース）を列挙する
- **UI（ユーザインターフェース）設計**
  抽出した機能をユーザの使い勝手と端末（iPhoneまたはiPad）の特性に合わせてグループにまとめてどのように見せるか決める

最終的にグループにまとまった機能を見せる単位が、画面として実現され、分割された画面間の行き来が画面遷移として実現されます。

2-1-2 アプリ開発の3大要素

機能が洗い出されてある程度アプリの画面や画面遷移が決まると、次は機能をプログラムに落とし込んで実装していきます。開発者は実装すべき機能を「機能を見せる」「機能をつなげる」「機能を実現する」の3つのパートに分けてプログラムしていきます。図にすると、以下のような感じです。

▼ 図2-1-2　アプリ開発の3大要素



「機能を見せる」「機能をつなげる」「機能を実現する」の3つについて詳細を以下に述べます。

- **機能を見せる**
  ラベル、ボタン、テキスト入力など画面を構成するための部品（UI部品またはビューと呼ぶ）とその組み合わせを実装する
- **機能をつなげる**
  ビュー同士の動きの連携や画面遷移、まとまった機能をつなげるなど、アプリの流れにかかわる部分を実装する
- **機能を実現する**
  情報を細かくデータ分割して、処理とデータをまとめる。データの取得や保存、計算、ロジックなどを実装する

上記の3つのパートは、Model-View-Controller（MVC）アーキテクチャというソフトウェア構造のパターンと対応しています。

- **機能を見せる　＝ View（ビュー）**
- **機能をつなげる　＝ Controller（コントローラ）**
- **機能を実現する　＝ Model（モデル）**

iOSアプリの画面と画面遷移の実装は「機能を見せる」と「機能をつなげる」と関係があります。その役割を担うのがUIViewとUIViewControllerクラスです。

残りの「機能を実現する」は、データ設計と関係があります。データ設計については第5章で詳しく説明します。

COLUMN

MVC

MVCとはModel-View-Controllerの略で、GUIでプログラムを設計するときの指針となるデザインパターンの一種です。もともとは、SmalltalkでGUIアプリケーションを開発するための設計指針として生み出されたパターンです。iOSの元になったCocoaフレームワークがMVCのアーキテクチャを基本構造に取り入れているため、iOSもMVCに基づいて設計されています。

■MVCの役割

MVCでは、プログラムを「モデル」「ビュー」「コントローラ」の3つのパーツに分けて開発していきます。

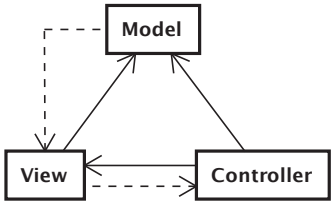
モデルは情報（データ）を扱うオブジェクトです。住所管理アプリの名前や住所といったデータ

を格納するオブジェクトがモデルになります。

ビューはアプリの見た目を表現するオブジェクトです。ボタンやスイッチ、ラベル、テキスト入力、画像表示など、ユーザが直接目で見る部分を構築します。また、モデルに保存されているデータを画面に表示する役割も持っています。

コントローラはユーザからの入力を受けてモデルやビューを操作したり、画面遷移を制御したりするオブジェクトです。アプリ全般にかかわる処理をコントローラで行います。MVCの構造を図にすると、以下のようになります。

▼ MVCの構造



モデルはビューやコントローラから参照されていますが、モデル自体はビューやコントローラを参照していません。モデルを独立させることで、複数の画面で同じモデルを使用することができます。ビューは間接的にコントローラを参照することがあります。ビューの状態が変わったことをコントローラに知らせたり、ユーザの入力をコントローラに知らせたりするために参照します。iOSではデリゲートパターンを使ってビューからコントローラへ参照することが多いです。

iOSから提供されているクラスをMVCに分類すると、以下のようになります。ただしすべてのクラスが「モデル」「ビュー」「コントローラ」の3つに分類されるわけではありません。文字列操作を支援したり配列操作を支援するなど、MVCに属さないクラスも存在します。

モデル

NSManagedObjectクラス（Core Data）とそのサブクラス

ビュー

UIViewとそのサブクラス

コントローラ

UIViewControllerとそのサブクラス、NSFetchedResultsControllerクラス（Core Data）など

コントローラがあるおかげでビューやモデルの独立性が高くなり、ビューとモデルの再利用性が高まります。

ビューとコントローラについては本章で詳しく説明しています。また、モデルについては第3章で詳しく説明します。

2-1-3 iOSでどんなアプリを開発できるのか？

それでは、iOSでどのようなアプリを開発することができるのかを見てみましょう。iOSで開発することができるアプリは、以下のように分類できます。

- **Single View Application**
  画面が1つだけのシンプルなアプリ
- **Utility Application**
  メイン画面と設定画面で構成されるアプリ
- **Master Detail Application**
  一覧画面と詳細画面で構成されるアプリ
- **Tabbed Application**
  タブを使って複数画面を切り替えるタイプのアプリ
- **Page-Based Application**
  ページめくりを使うアプリ
- **OpenGL Game**
  OpenGLを使ったゲーム。ゲームについては本書の範囲外なので、詳細は割愛

アプリ開発をはじめるにあたって、開発者は上記アプリのうち1つを選んでベースにします。そのベースをもとに実現したい機能とマッチする画面と画面遷移を選択します。iOSアプリで使われる画面や画面遷移にもいくつかの種類があり、これらパターンを組み合わせてアプリを開発していきます。

以下は、iOSアプリの代表的な画面と画面遷移の種類を分類したものです。

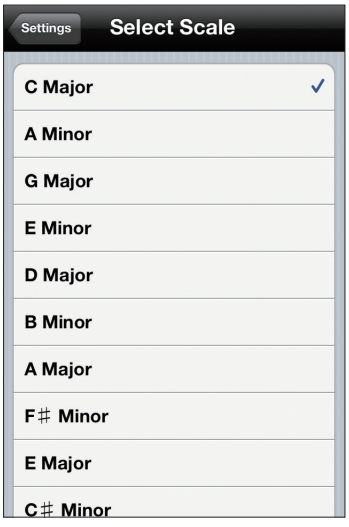
▶ 画面の種類

画面の情報をリスト形式で表示するかしないかで、画面の種類が次のように分かれます。

- **リスト形式画面**

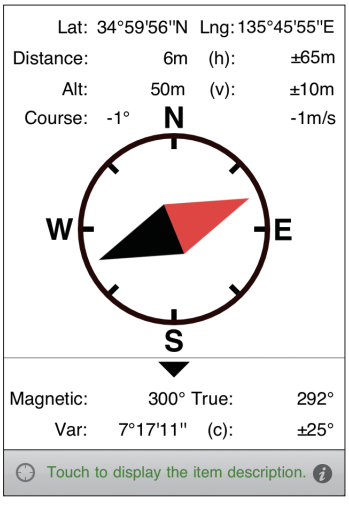
テーブルビューやコレクションビューを使ってリスト形式でデータを表示する画面。テーブルビューは複数のデータを縦方向にリスト形式で並べることのできるビューで、コレクションビューは複数のデータを縦横両方向に並べることができるビューのことです。

▼ 図2-1-3 リスト形式画面の例



- **ノーマルビューベース画面**
  テーブルビューやコレクションビューを使わない画面

▼ 図2-1-4 ノーマルビューベース画面の例



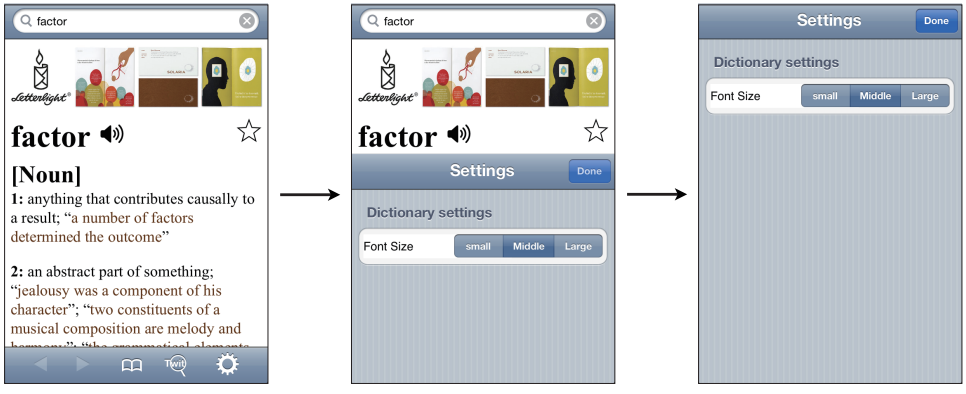
画面遷移の種類

一般的に画面遷移と言えば、ある画面から別の画面に移る（＝遷移する）ことです。それだけだと少し分類しにくいので、画面遷移を大きくとらえて画面の上に別の画面を

重ねて表示することや複数の画面を切り替えることも、画面遷移としています。iOSアプリの画面遷移の種類を分けると、以下のようになります。

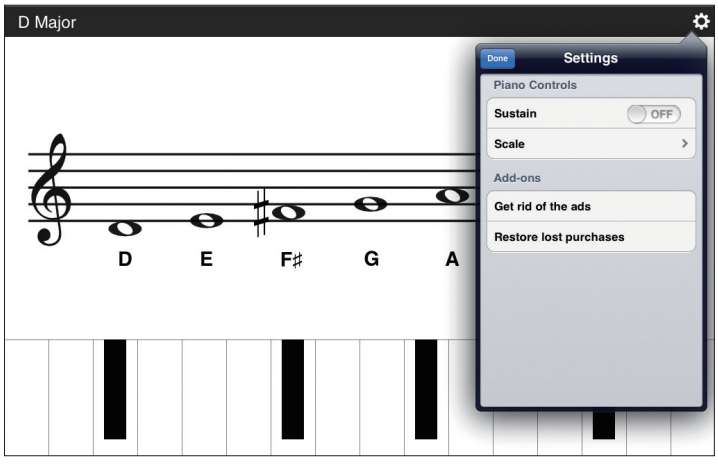
- **モーダル（Modal）**
  画面の上に新たに画面を重ねていく画面遷移方法。重ねられた（下位にある）画面は新たに開いた画面が閉じるまで操作できない。いちばん上に表示されている、ユーザが操作可能なビューのことをモーダルビューと呼ぶ

▼ 図2-1-5　モーダルの例



- **ポップオーバー（Popover）**
  画面の上に小さな画面を重ねて表示する。機能的にはモーダルと同じ。iPadで使用される

▼ 図2-1-6　ポップオーバーの例



- **ナビゲーション（Navigation）**
  階層構造のあるデータを詳細画面に掘り下げていく画面遷移方法。ドリルダウンと呼ばれることもある。テーブルビューとセットで使用することが多い

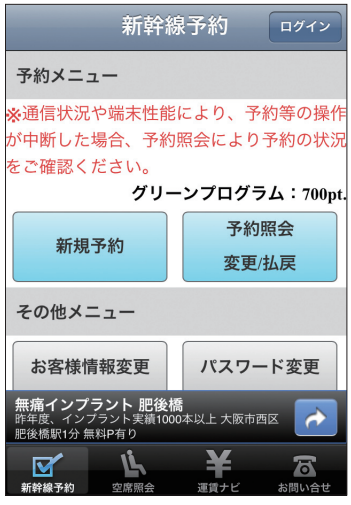
▼ 図2-1-7　ナビゲーションの例

- **スプリット（Split）**
  画面を2つの領域に分けて表示する。左側の項目を選択すると右側のページが切り替わる。iPadでナビゲーションの代わりに使用する。横向き（Landscape）と縦向き（Portrait）で表示が変わる。縦向きのときは右側の画面のみ表示され、左側の画面はポップオーバーで表示する

▼ 図2-1-8　スプリットの例

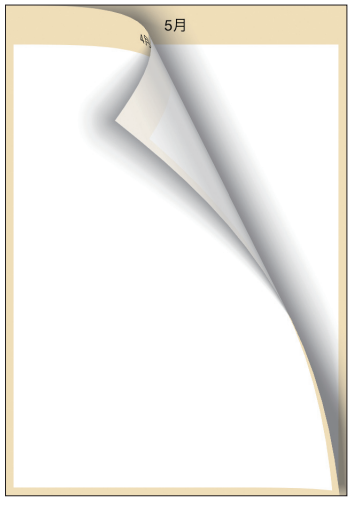
- **タブ（Tab）**
  タブを使って画面を切り替える画面遷移方法

▼ 図2-1-9　タブの例



- **ページめくり**
  電子書籍などページをめくって画面を移動する画面遷移方法

▼ 図2-1-10　ページめくりの例



2-1-4 iOSアプリにおける画面とは

iOSにおいて画面とは一体なにか、どんな構造をしているか見ていきたいと思います。

iOSアプリの画面はユーザが実際目にするビューの集まりと、それらを制御する1つのビューコントローラで構成されています。iOSでは、ビューコントローラとビューを含めた1つの画面のことをシーン（Scene）と呼びます。

画面 ＝ シーン ＝ ビューの集まり ＋ ビューコントローラ

iOSアプリでは、複数のビューが重なり合って画面の見た目を構成しています。ビューの実体はUIViewクラス（またはそのサブクラス）のオブジェクトです。また、画面に配置されたビューはツリー構造になっていて、ビューコントローラが管理しています。ビューコントローラの実体はUIViewControllerクラス（またはそのサブクラス）のオブジェクトです。

2-1-5 iOSアプリにおける画面遷移とは

iOSにおいて画面遷移とはいったいなにかを見ていきたいと思います。

iOSでは、ユーザに見えているビューを管理するビューコントローラから別のビューコントローラに制御を移して別のビューを見せることで、画面遷移を実現しています。ビューコントローラの遷移が画面遷移になります。iOSでは、ビューコントローラの遷移のことをセグエ（segue）と呼びます。

segue（セグエ）という聞き慣れない単語が出てきますが、これは「ある状態から次の状態へ滑らかに移行する」という意味の動詞です。推測ですが、少し前に話題になった乗り物セグウェイ（Segway）は、segueが語源だと思います。

画面遷移 ＝ セグエ ＝ ビューコントローラの遷移

これ以降、アプリの画面説明では単に画面遷移と表現しますが、プログラム説明のときにはビューコントローラの遷移またはセグエという表現を使います。

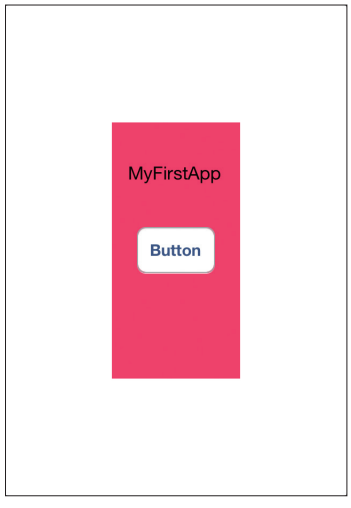
2-1-6 iOSアプリの画面と画面遷移の作成方法

iOSアプリの画面と画面遷移を作成するには、Xcodeに付属しているストーリーボードと呼ばれるツールを使います。

ストーリーボードを使うと、ビューの配置やビューコントローラの遷移（セグエ）をプログラムを書かずビジュアルに作成することができます。では実際に、ストーリーボードを使って画面をデザインする方法を見てみましょう。

以下の簡単なアプリを例に、画面と画面遷移を作成していきます。

▼ 図2-1-11　サンプルアプリの画面イメージ

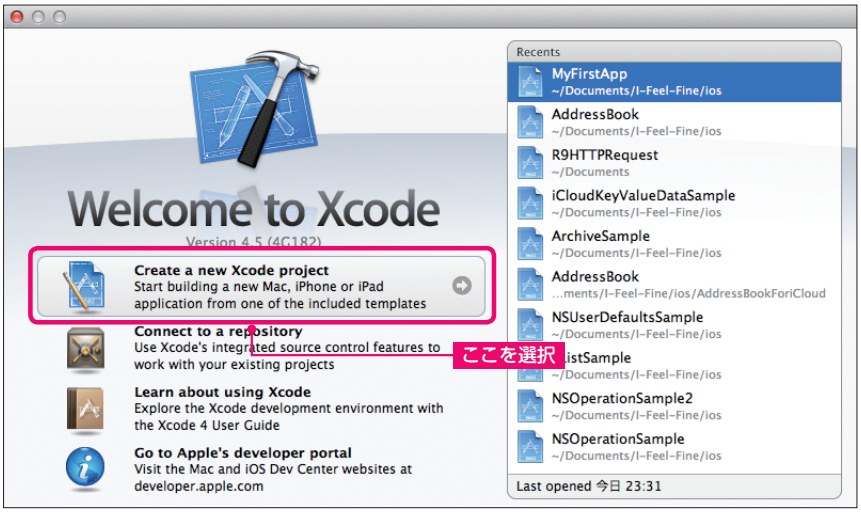


画面いっぱいに表示したUIViewオブジェクトの上に適当な大きさのUIViewオブジェクトを重ねて、さらにその上にUILabelオブジェクトとUIButtonオブジェクトを配置しています。

プロジェクトの作成

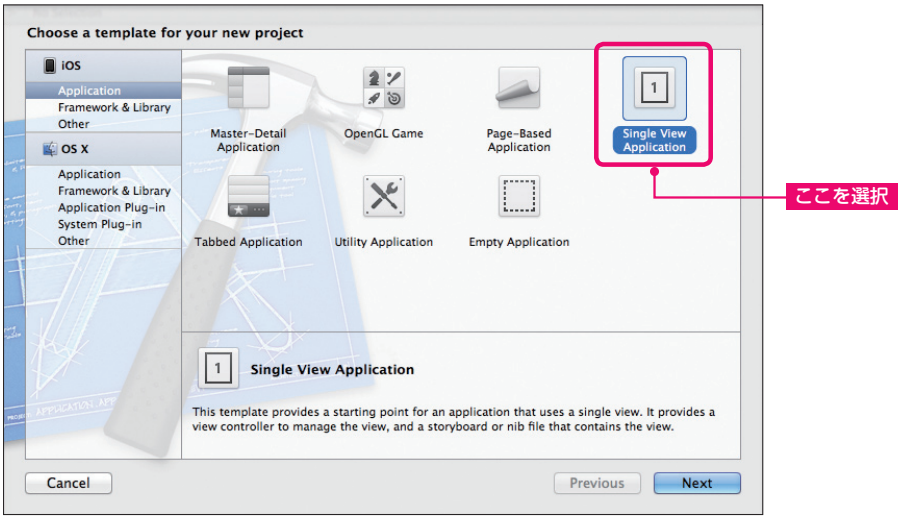
まずはじめにプロジェクトを作成します。Finderの「アプリケーション」からXcodeを立ち上げて「Create a new Xcode project」を選択して「Open」を押してください。

▼ 図2-1-12 Xcodeの起動画面イメージ



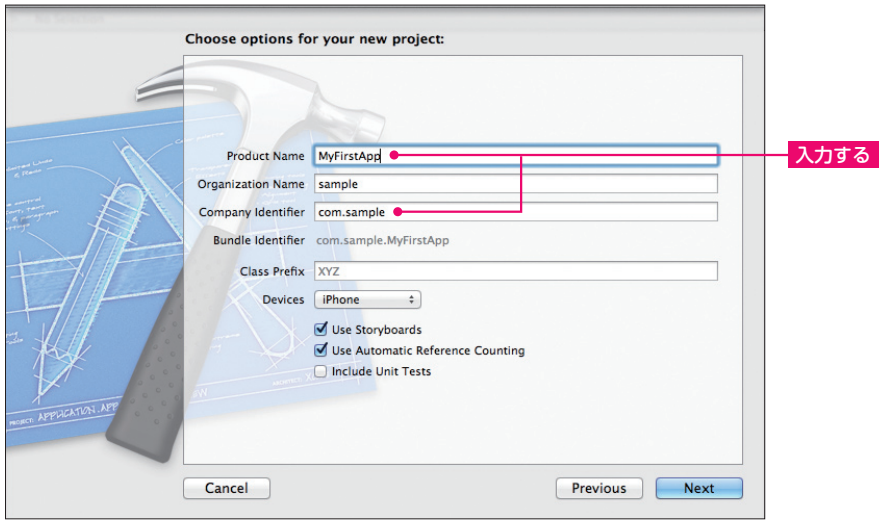
次に「Single View Application」を選択して、「Next」ボタンを押してください。

▼ 図2-1-13 アプリの種類を選択する画面イメージ



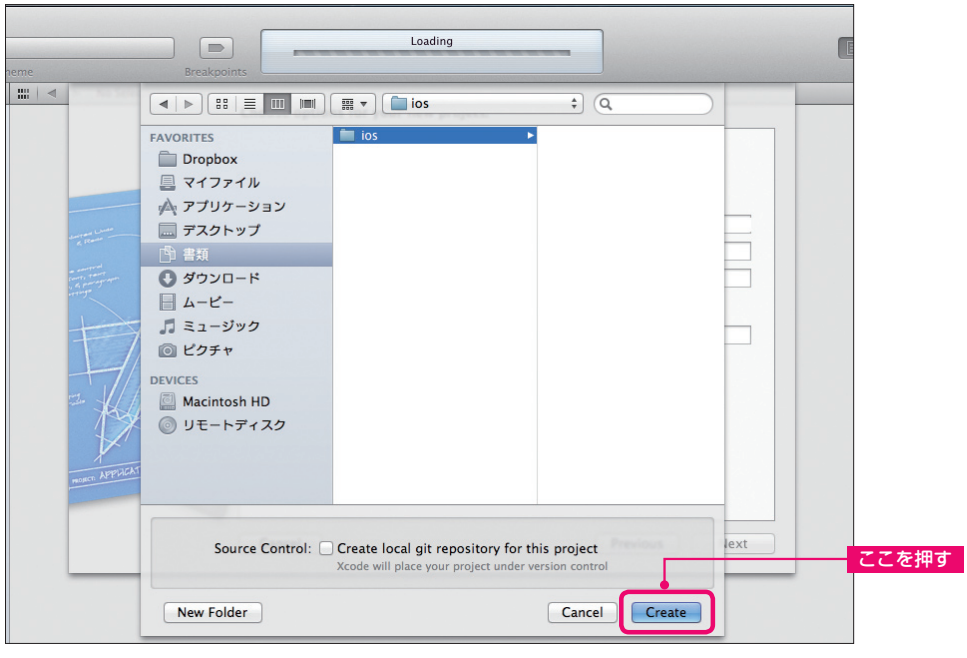
次に、「Use Storyboards」と「Use Automatic Reference Counting」にチェックが入っていることを確認して、「Product Name」と「Company Identifier」を入力後、「Next」ボタンを押してください。例ではそれぞれMyFirstApp、com.sampleと入力しています。また、DevicesはiPhoneを選択してください。

▼ 図2-1-14　Product NameとCompany Identifierを入力する画面イメージ



最後に、プロジェクトの保存場所を指定して「Create」ボタンを押してください。例では「書類」フォルダの中に「ios」フォルダを新たに作成して保存しています。環境に合わせて好きな場所に保存してください。

▼ 図2-1-15　プロジェクトの保存場所を指定する画面イメージ

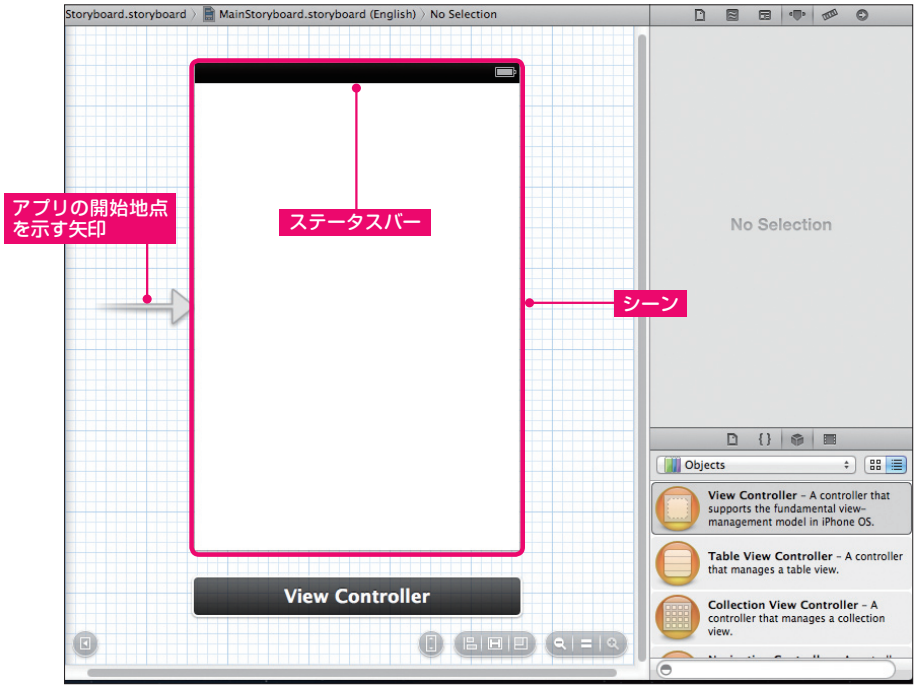


画面（シーン）にビューを配置する

アプリの画面（シーン）にビューを配置していく方法を見ていきましょう。

Xcodeの Project Navigatorから MainStoryboard.storyboardファイルをクリックして、ストーリーボードを開きます。

▼ 図2-1-16　シーンが表示される



それでは、シーンにビューを配置してみましょう。まずはじめに次のように、右下にある Object Libraryの「View」をマウスで選択して、シーンの上にドラッグします※1。

※1：Object Libraryが表示されない方は、Xcodeメニューの「View > Utilities > Show Object Library」を選択すると表示されます。