



G's ACADEMY

Dev16 iOS day4

2020.6.6

Naoki Kameyama

わほーい



iOSアプリ開発は、

1

直感的！

2

応用が効く！

3

楽しい！

今日やること

1

Appleが用意したDelegateメソッドに慣れよう

- Delegateは”双方向”でメソッドを呼びたいときに用いる
- リアルタイム文字数カウントアプリ

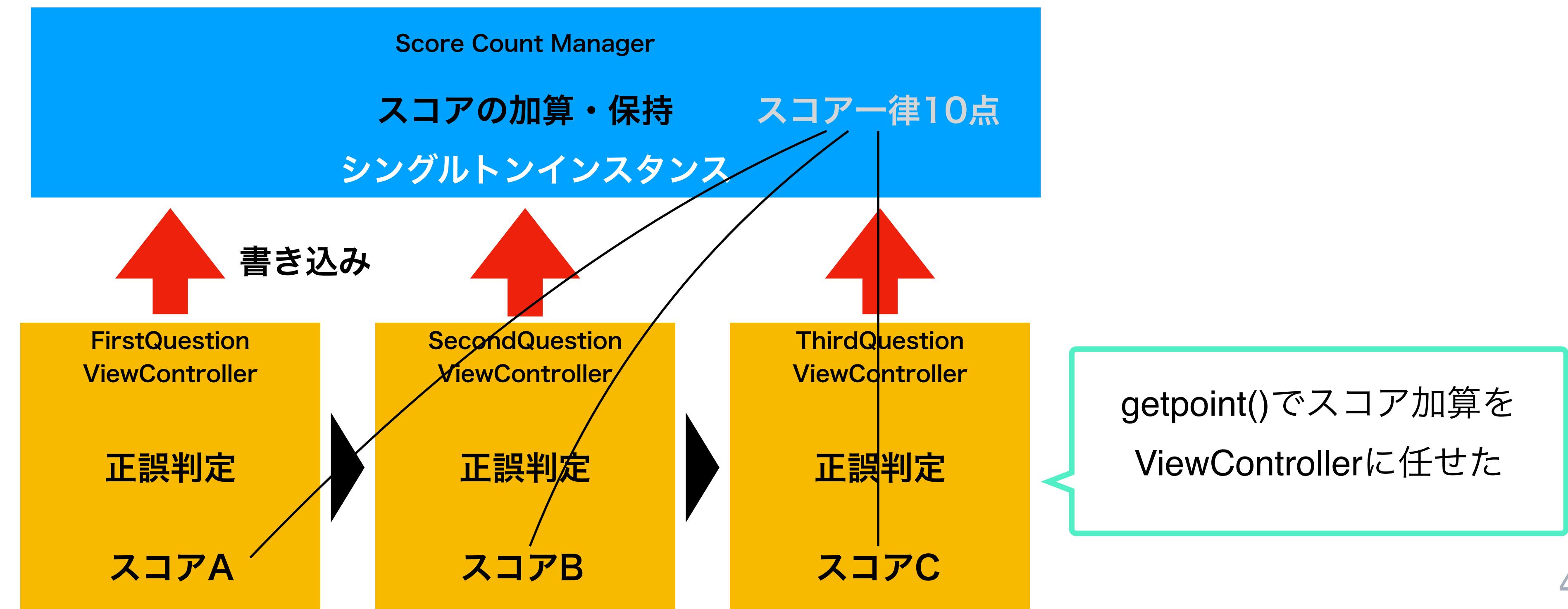
2

TableViewを使ってみよう！

- TableViewとCell
- SectionとRow
- customでTableViewを好きな様につくってみよう

デリゲートデザインパターンについて

- ・特定の処理や値を、別のクラスに「まかせる（委譲）」仕組み
- ・protocolを活用してデリゲート元のクラス→デリゲート先のクラスにまかせる
- ・デリゲート先のクラスはprotocolを準拠していれば、複数、異なるクラスで利用可能
- ・デリゲート先にまかせるデリゲートメソッドの処理は、それぞれカスタマイズできる



Delegateをどんな時に利用するか(自分が理解したイメージ)

2

通知を使ったアプリをつくってみよう

- ・ クイズアプリの作成
- ・ Delegateを仕組みを知る

"通知を使った"？
どうゆうこと？



Delegateをどんな時に利用するか(自分が理解したイメージ)

Class ViewController

```
class ViewController: UIViewController {  
  
    let fuga: FugaFuga = FugaFuga()  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        // FugaFugaのインスタンスのメソッドを呼び出せる  
        fuga.printFuga()  
    }  
  
}
```

Class FugaFuga

```
class FugaFuga {  
  
    let value = "FugaFugaのvalueだよ"  
    func printFuga() {  
  
        print("printFugaが実行されたよ🐱")  
    }  
}
```

Class ViewController

FugaFugaのインスタンスを保有
= "FugaFugaに依存している"

好きなときにFugaFugaの
メソッドを呼び出せる



Class FugaFuga

好きなときにViewControllerの
メソッドを呼び出せない

Delegateをどんな時に利用するか(自分が理解したイメージ)

Class ViewController

```
class FirstQuestionViewController: UIViewController, ScoreDelegate {  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        //デリゲート先が自分自身であることを宣言  
        ScoreCountManager.sharedInstance.delegate = self  
    }  
  
    //ScoreDelegateプロトコルに準拠したメソッド（デリゲートメソッド）  
    func getPoint() -> Int {  
        return 5  
    }  
}
```

Class ScoreCountManager

```
class ScoreCountManager{  
  
    //まかせる先のクラス  
    weak var delegate:ScoreDelegate?  
  
    //得点を追加する  
    func addScore(){  
        let result = self.delegate?.getPoint()  
        print(result!)  
    }  
}
```

Class ViewController

ScoreCountManagerは
シングルトンなのでどこでも使える

好きなときにaddScore()の
メソッドを呼び出せる

Class ScoreCountManager

Delegateを使って
好きなときにgetPoint()の
メソッドを呼び出せる

ViewControllerへの”通知”

つまり

Delegateとは、

コードの書き方は特殊だけど、

通常、片方向でしか好きなときにfuncを呼べなかつたけど

双方向で好きなときにfuncを呼べるようになる

便利な機能なんだ。

コードの書き方はググればOK,

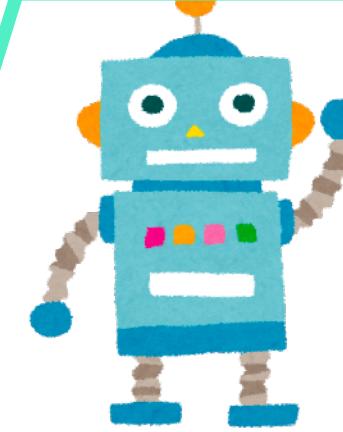
大事なのはどんな時に使うと便利かを知ること！



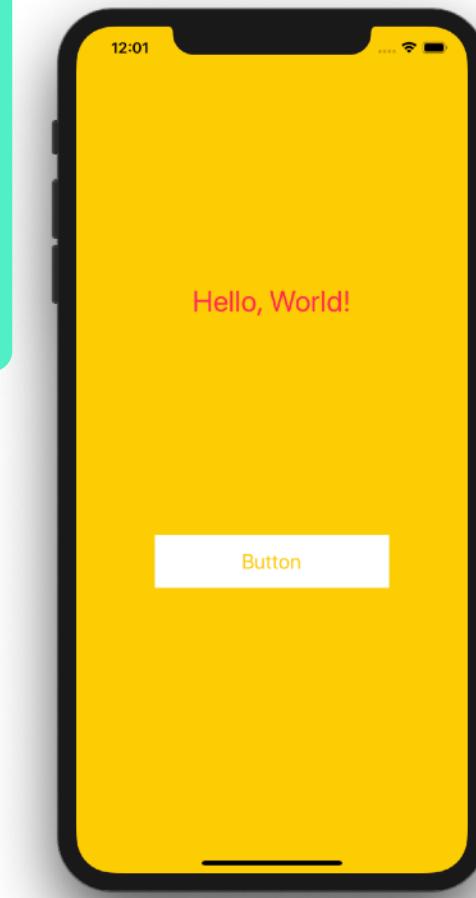
Appleが用意しているDelegate

Class ViewController

画面全体のパーツや変数を管理し、
画面表示タイミングで
動作を決めるぜ！

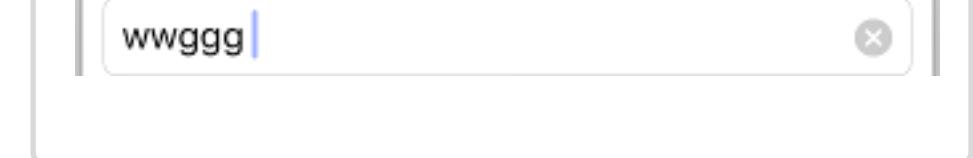


ViewController



Appleが用意しているClass

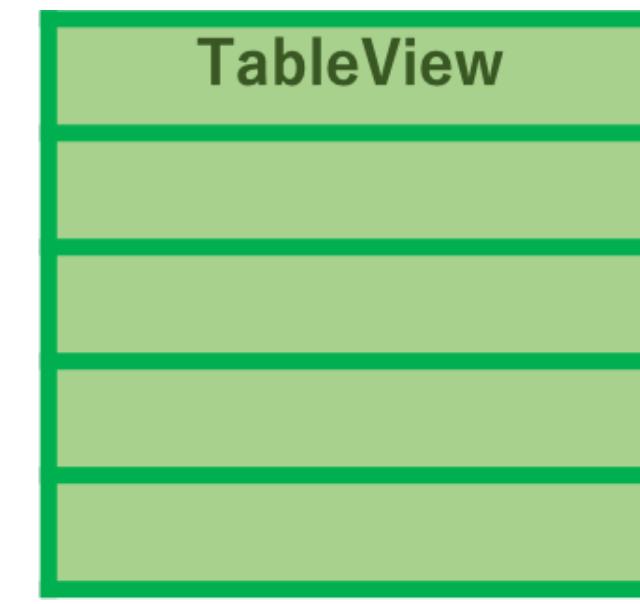
UITextField



UITextView

UITextViewUITextView
UITextViewUITextView
UITextViewUITextView

UITableView



Class ViewController

好きなときに
メソッドを呼び出せる



Appleが用意しているClass

Delegateを使って
好きなときにAppleが決めた
メソッドを呼び出せる

例： UITextViewを編集が
完了したとき

例：UITextViewのDelegateメソッド

UIKitのDelegateメソッドはAppleが決めているので、自分で勝手に作ることはできません。ドキュメントを調べましょう。

Responding to Editing Notifications

func `textViewShouldBeginEditing(UITextView) -> Bool`

Asks the delegate if editing should begin in the specified text view.

func `textViewDidBeginEditing(UITextView)`

Tells the delegate that editing of the specified text view has begun.

func `textViewShouldEndEditing(UITextView) -> Bool`

Asks the delegate if editing should stop in the specified text view.

func `textViewDidEndEditing(UITextView)`

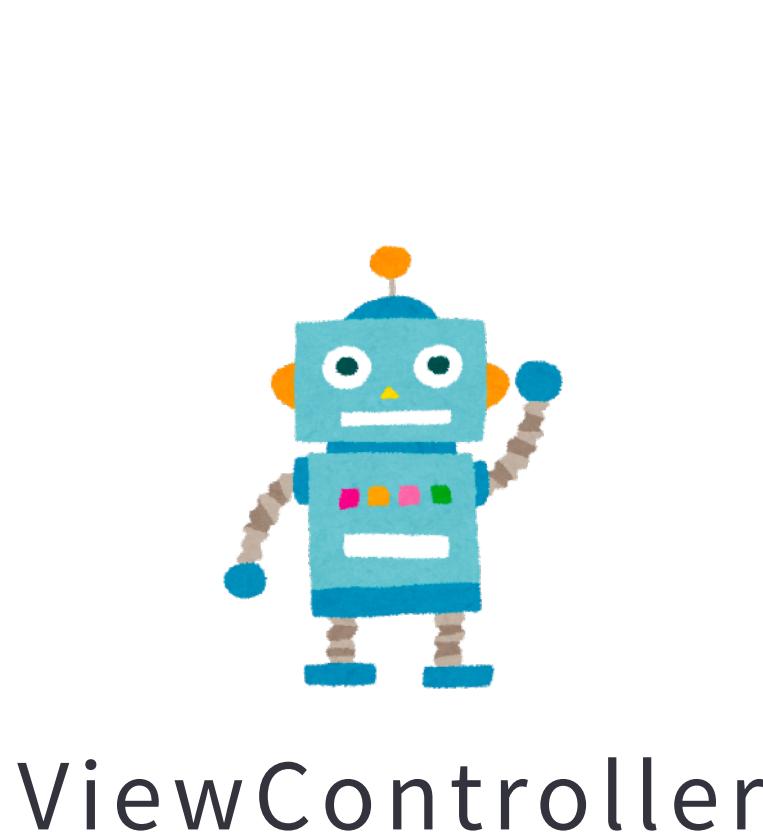
Tells the delegate that editing of the specified text view has ended.

他にもあります。調べてみてください。

<やってみよう>
リアルタイム文字数
カウントアプリを作る

UITextView の変更を通知するDelegateメソッド

Class ViewController



Appleが用意しているClass

UITextView

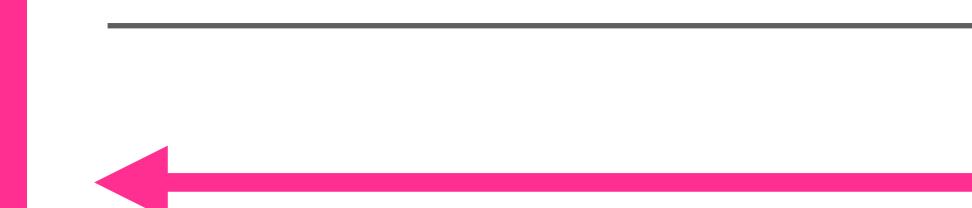
UITextViewUITextView
UITextViewUITextView
UITextViewUITextView

Delegate メソッド

func `textViewDidChange(UITextView)`

textViewが変更されたときに呼ばれるメソッド

Class ViewController



Class UITextView

Delegateを使って
textViewが変更されたときに
メソッドを呼び出せる

textViewが変更されたとき

1. UIパーツを配置



細かな値は適当で良いです！

TextView

上左右 画面端から 48pt

Height: 300pt

UILabel

左右中心 UITextViewとの隙間 50pt

UILabel

左右中心 UITextViewとの隙間 30pt

fontの大きさ: 63

2. ViewControllerと接続

```
8  
9 import UIKit  
10  
11 class ViewController: UIViewController {  
12  
13     @IBOutlet weak var textView: UITextView!  
14     @IBOutlet weak var textCountLabel: UILabel!  
15  
16     override func viewDidLoad() {  
17         super.viewDidLoad()  
18         // Do any additional setup after loading the view.  
19     }  
20  
21 }
```

-UITextView
-UILabel

2点をOutlet接続

3. Delegateを使う準備

①protocolの追加

```
class ViewController: UIViewController, UITextViewDelegate {  
  
    @IBOutlet weak var textView: UITextView!  
    @IBOutlet weak var textCountLabel: UILabel!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        // Do any additional setup  
  
        // 読み方: textViewからコードを任される(通知される)のは、self(このViewController)ですよ  
        textView.delegate = self  
    }  
  
    func textViewDidChange(_ textView: UITextView) {  
        print("textViewDidChangeが呼ばれたよ")  
    }  
}
```

②役割を任されたクラスを
設定する

③Delegateメソッドを
書く

Delegateはいつでも3ステップ！

3. Delegateメソッド内にcountするロジックを追加

```
// TextViewを変更したときに呼ばれるDelegateメソッド
func textViewDidChange(_ textView: UITextView) {
    print("textViewDidChangeが呼ばれたよ")
    let count: Int = textView.text.count
    textCountLabel.text = "\(count)"
}
```

Textview.text.count で文字数を
取得できます。

4. (余談)キーボードを下げる方法

```
// TextView外がタップされたときに呼ばれるメソッド（もともとUIViewControllerがもっているメソッドを上書き）
override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?) {
    print("touchesBeganが呼ばれたよ")
    textView.resignFirstResponder()
}
```

1. textView以外をタップするしたときに呼ばれるtouchesBeganメソッド
2. textView.resignFirstResponder() で注目をresign(辞める)指示を出す

textViewを扱うときのお約束。

今日やること

1

Appleが用意したDelegateメソッドに慣れよう

- Delegateは”双方向”でメソッドを呼びたいときに用いる
- リアルタイム文字数カウントアプリ

2

TableViewを使ってみよう！

- TableViewとCell
- SectionとRow
- customでTableViewを好きな様につくってみよう



もでーん

おなかいっぱいなかめくん



ぐしゅん

泣き虫かめくん



いまいくの

いくフリしておっくうになってるかめくん



むりなの

拒否するかめくん



きゅうけい

さぼってるかめくん



にやーっ

驚いてへんな声だしてるかめくん



にしー

ほめてもらったかめくん

TableViewについて学ぶ

TableViewとは

アイテムのリストを单一の列に縦方向に表示
縦スクロールで見るもの

IndexPath

||

(section, row)

(0, 0)



(0, 1)

(0, 2)

(0, 3)

(0, 4)

(1, 0)

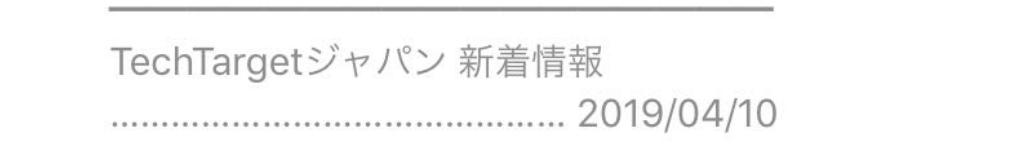
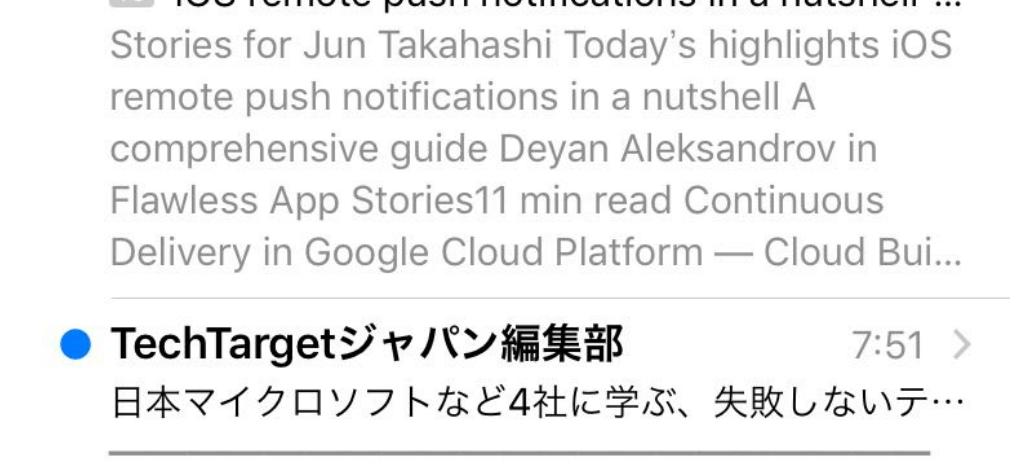
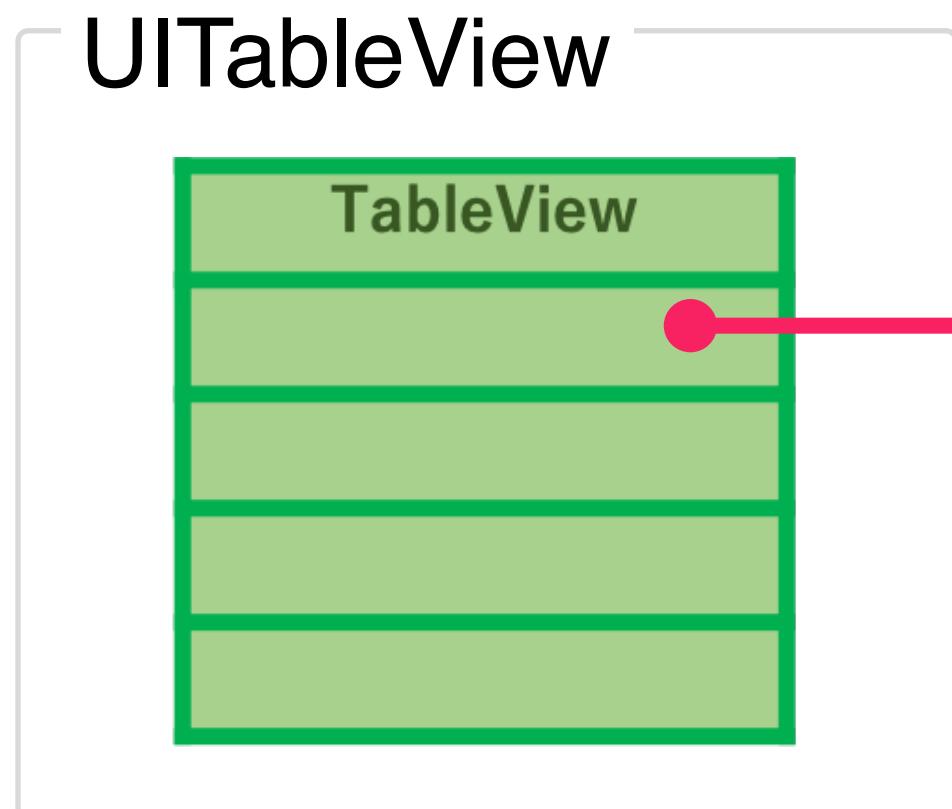
(1, 1)

(1, 2)

(1, 3)

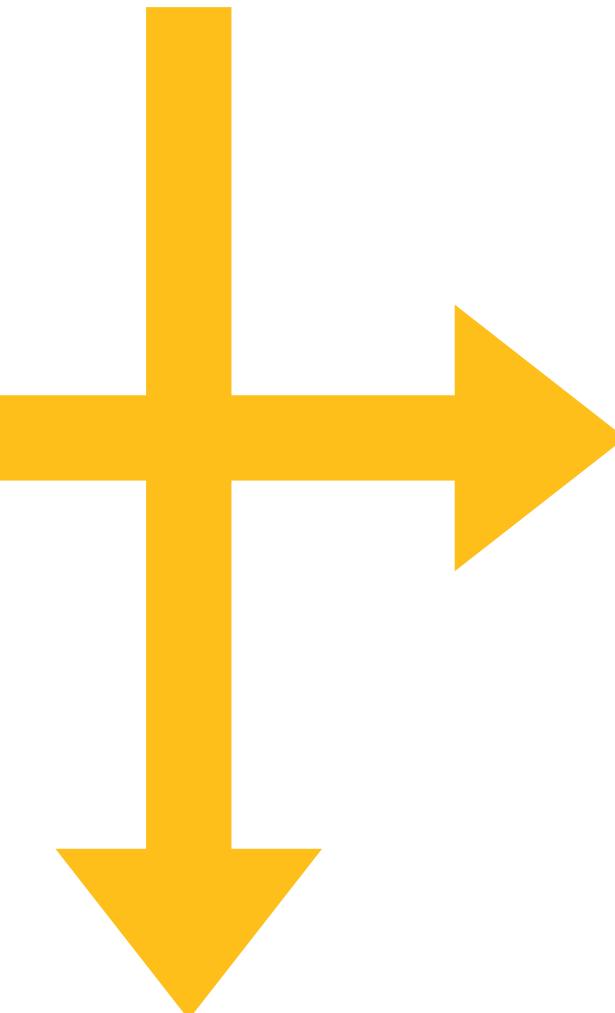
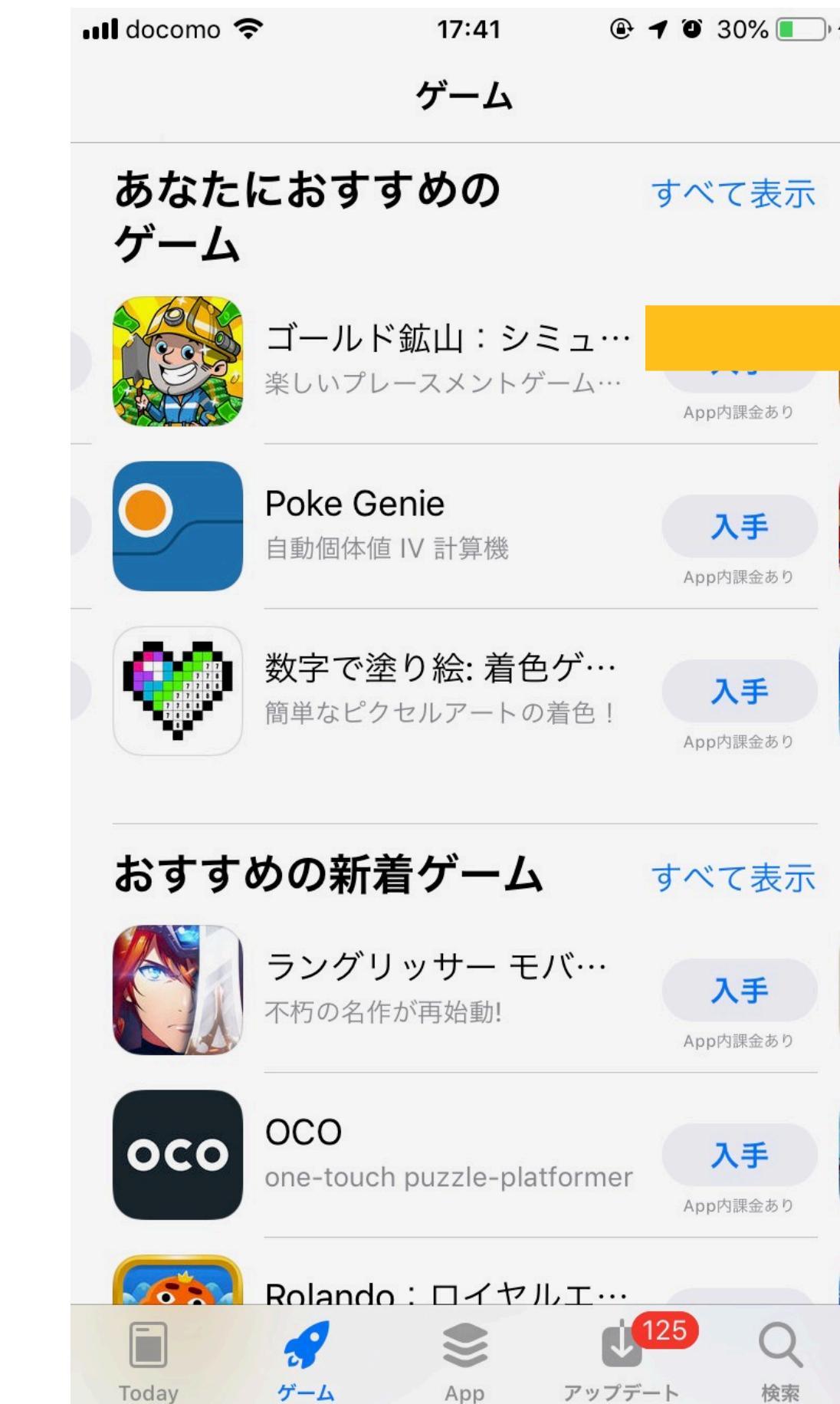
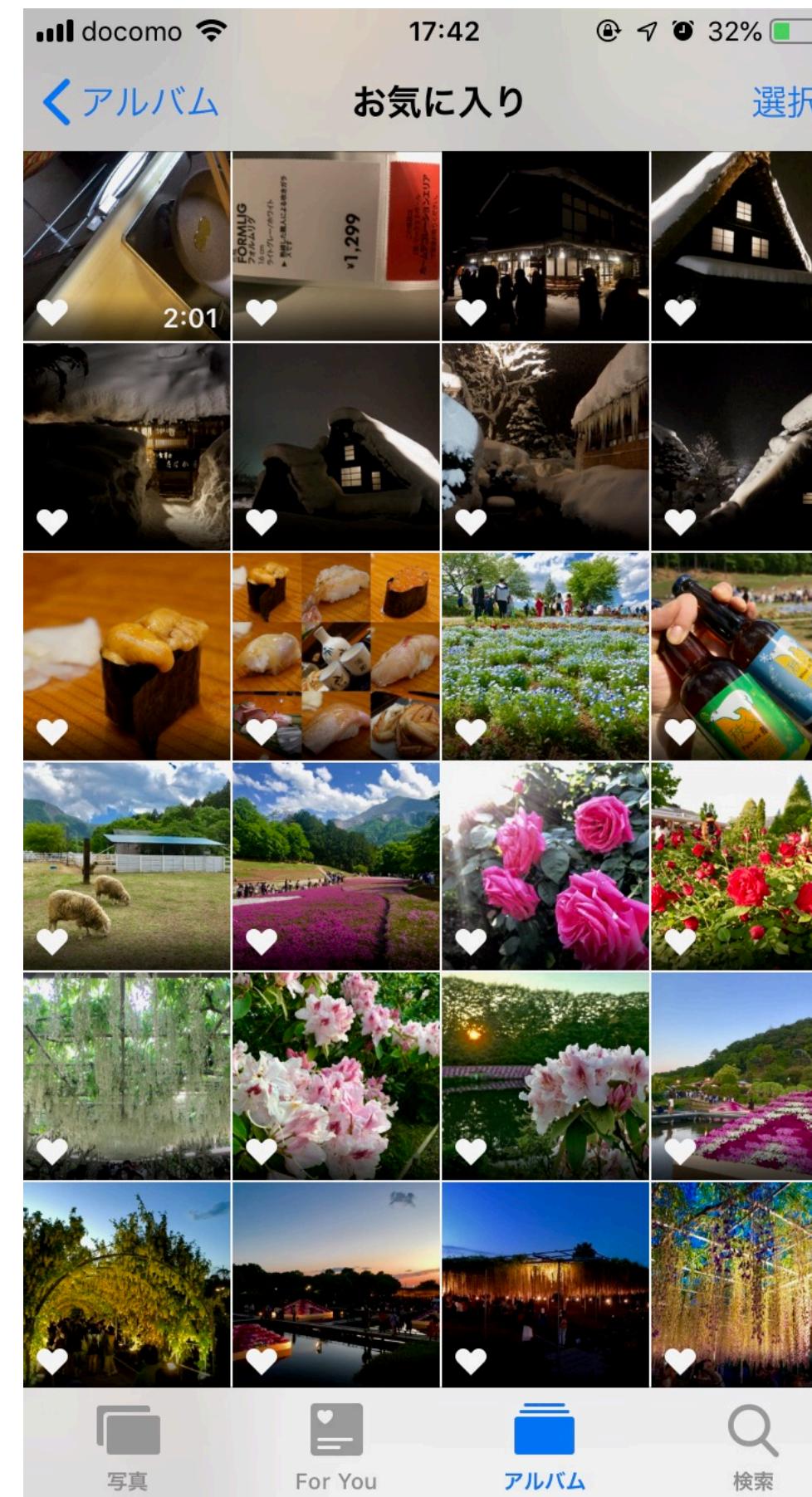
(2, 0)

(2, 1)



補足：CollectionViewとは

アイテムのリストを複数の行と列で持つことができて
縦および横スクロールで見るもの



TableViewを制すものは

iOSを制す



UIViewController + UITableView VS UITableViewController

TableViewを表示する場合以下の2つの選択肢があります

	UIViewController + UITableView	UITableViewController
概要	VC内にTableViewを手動で配置	最初からVCとTableViewが一体化
レイアウト	TableViewに対して 自由に制約をつけてレイアウト	最初から画面一杯に表示されていて、 制約を変更することができない
初期作業 の違い	IBOutletでの紐付けや、 VCに対するdelegate/datasource 設定が必要	IBOutletでの紐付けや、 VCに対するdelegate/dataSource設 定は不要（最初からできている）
StaticCell	使用不可 *	使用可能

少し複雑だけど自由

簡単だけど制約多い

補足：Static Cellとは



Static Cellは、主に設定画面のTableなど、
セルを静的に固定しておきたいケース
などで使用される

↔ ToDoアプリなどでは動的に
表示内容を変えたいのでStatic Cellを使用しない

Static Cellの場合、内容が固定なので、
DataSourceメソッドを書かずに
Storyboardでそのままセルの見た目を
設定することができる

また、セル内のアイテムをIBOutletで
直接VCに紐づけて設定することも可能

Prototype Cell VS Custom Cell

セルのレイアウトをIB上で作る場合、以下の2つの選択肢があります

	Prototype Cell	Xib
概要	<p><u>storyboard</u>を利用して TableViewの中にTableViewCell を配置して そのまま見た目やレイアウトを作 る</p>	<p>ViewやCellなどアプリ内部品のレ イアウトを デザインするための「Xib」ファイ ルを作って設定する</p>
メリット	<ul style="list-style-type: none">手数が少ないので設定がカンタ ン	<ul style="list-style-type: none">Cellを使い回してコードの肥大化 を防げる複雑な与件にも対応しやすい

こっちの方法

全体の手順（覚えておこう！）

Cell側の作業

1

UITableViewControllerのXibファイルとswiftファイルを追加

2

XibファイルでUIパーツを配置, Outlet接続

3

CellのID(identifier)を設定

全体の手順（覚えておこう！）

VC側の作業

1

ViewControllerにTableViewを配置してOutletで接続

2

Delegateを使う準備をする

- UITableViewDelegate, UITableViewDataSource のprotocol設定
- tableView.delegate = self, tableView.datasource = self

3

Cellの登録

- self.tableView.register(nib, forCellReuseIdentifier: cellId)

4

Delegateメソッドを書く

①Cell用のファイルを作成

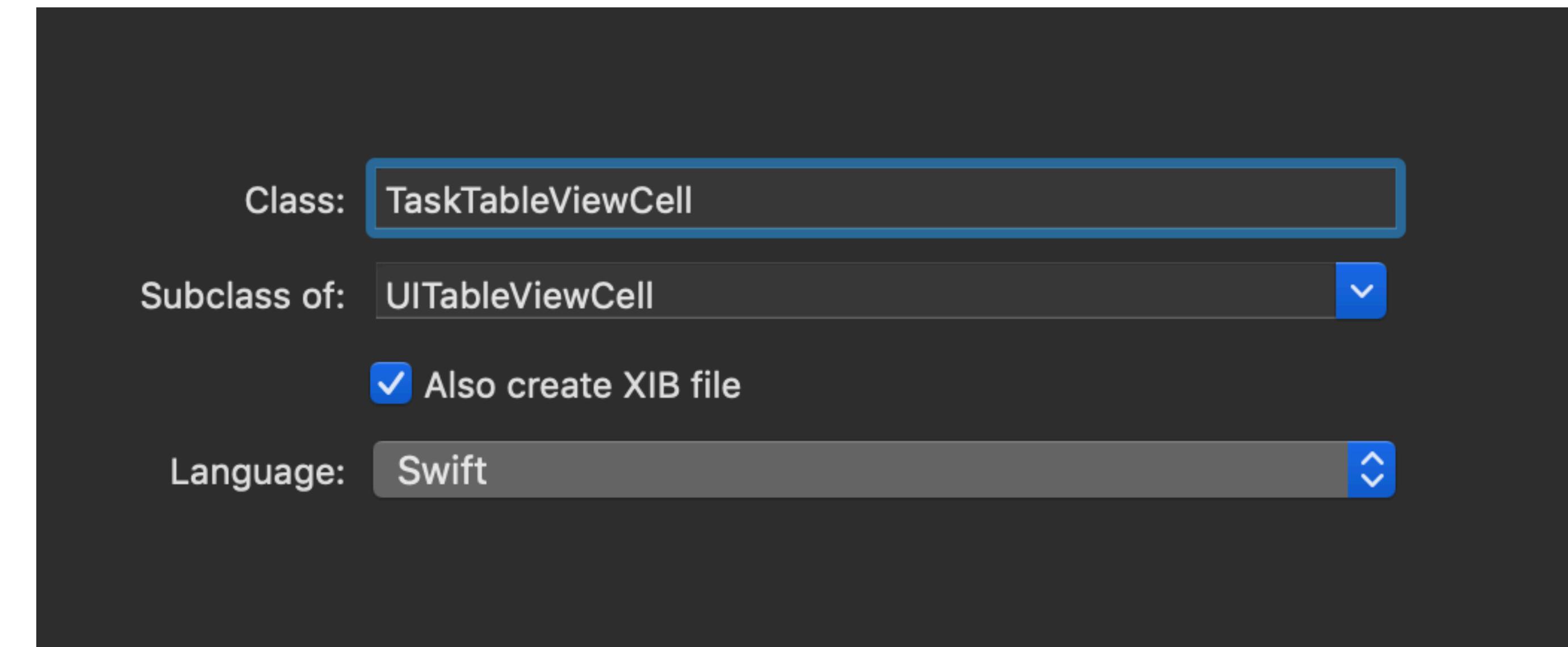
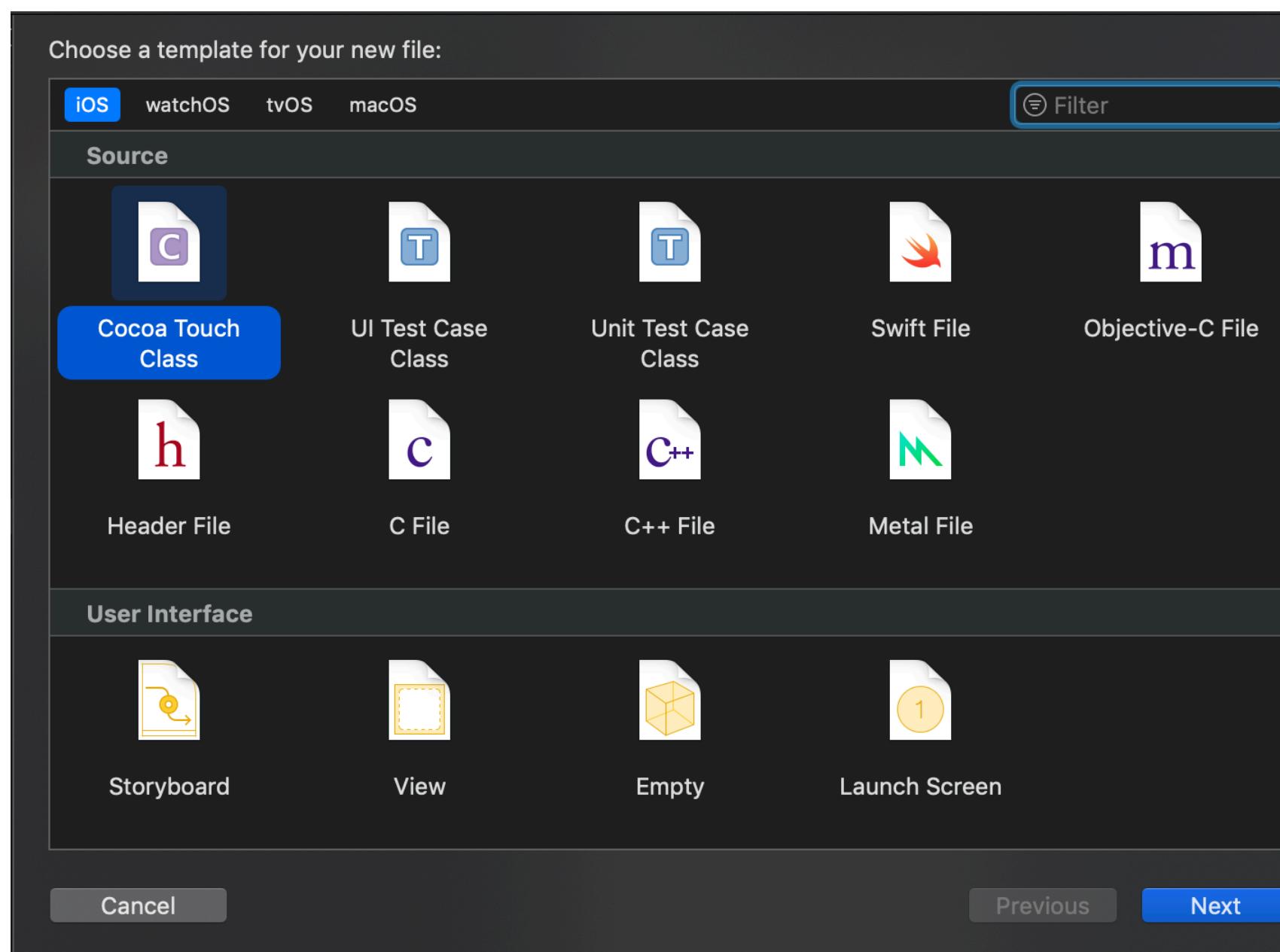
1-1. File > New > File > Cocoa Touch Class を選択

1-2. 「Subclass of」 欄で「UITableViewCell」を選択

1-3. 「Class」 欄でクラス名を入力したら

「Also create XIB file」をチェックして「Next」

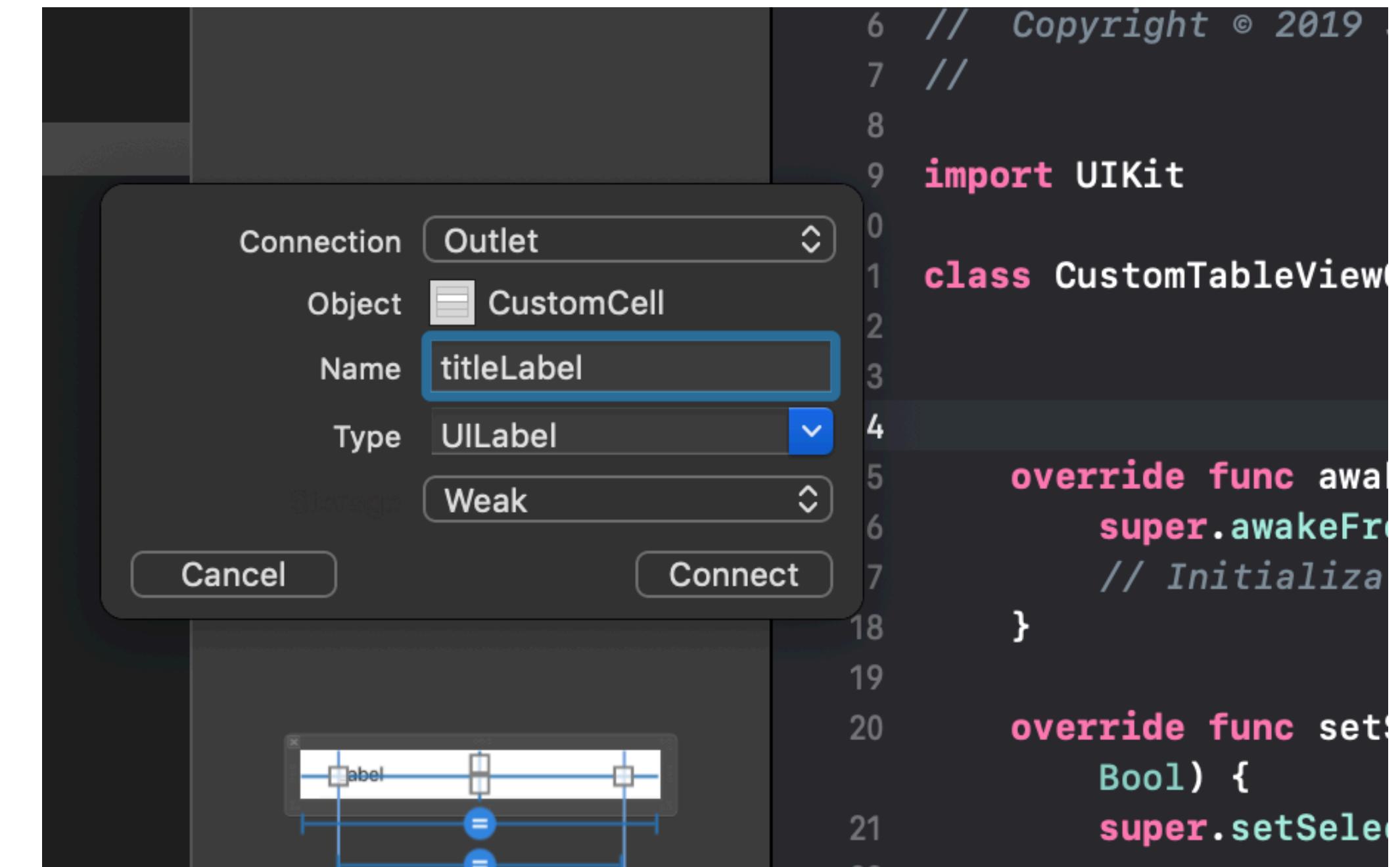
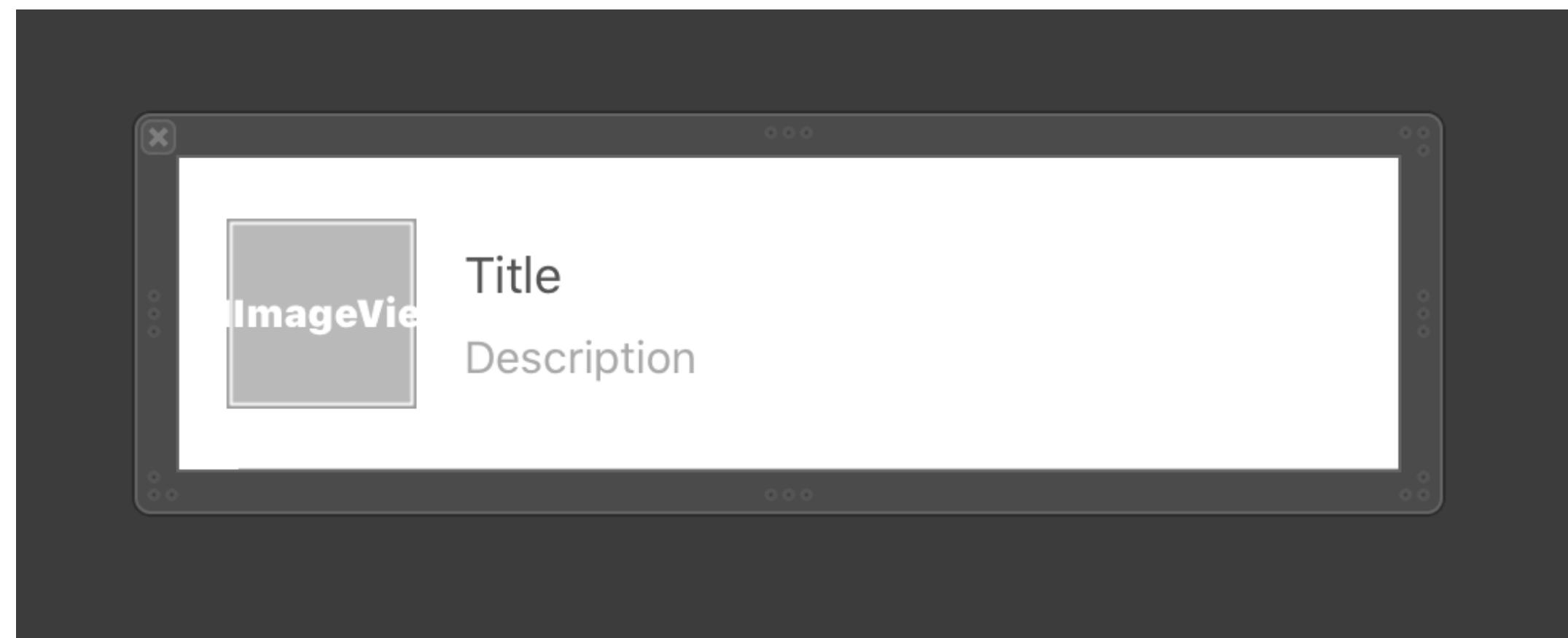
1-4. 保存場所を指定して「Create」



②Cellの見た目を設定

2-1. CellのXibを開いてUIパーツを配置

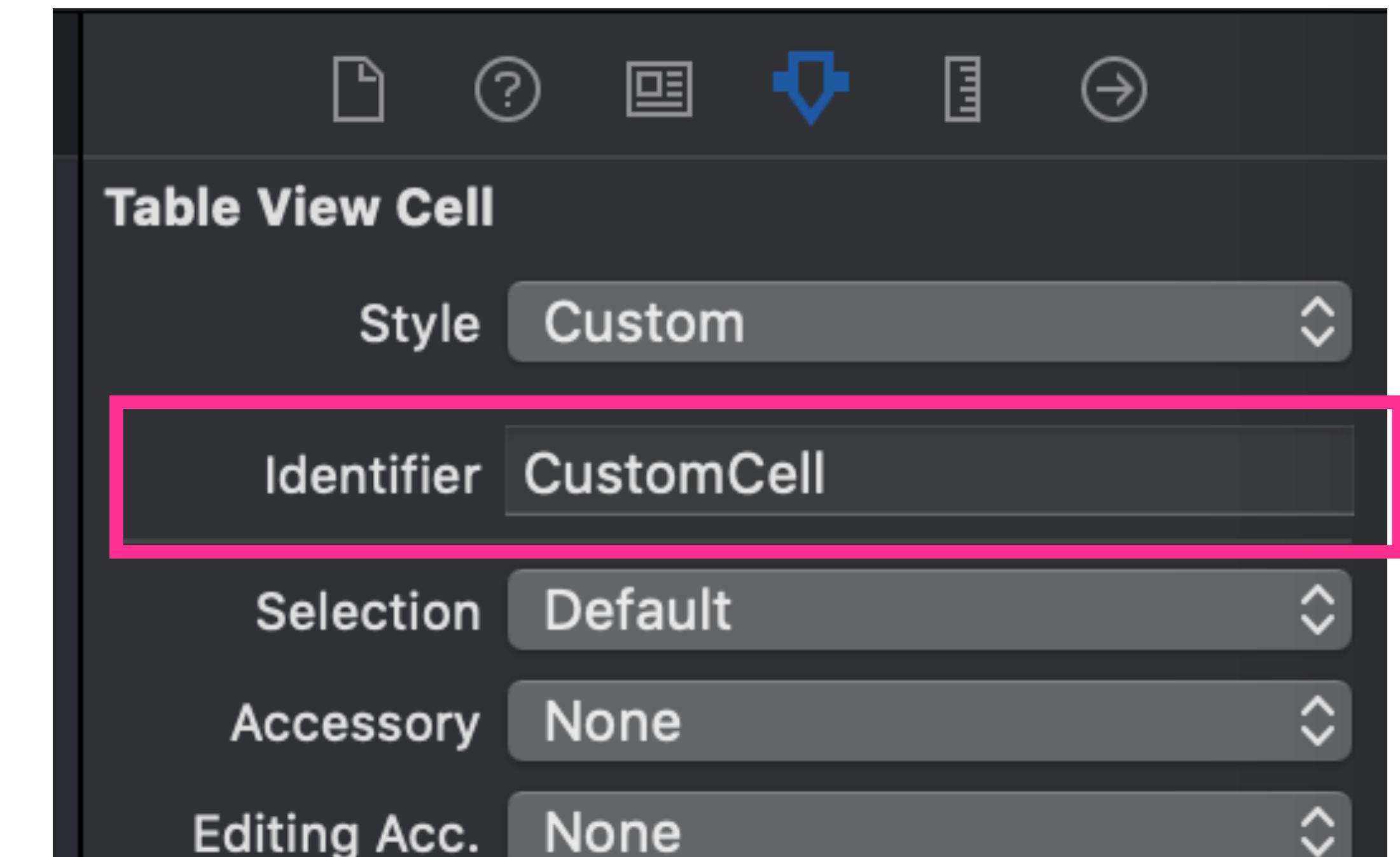
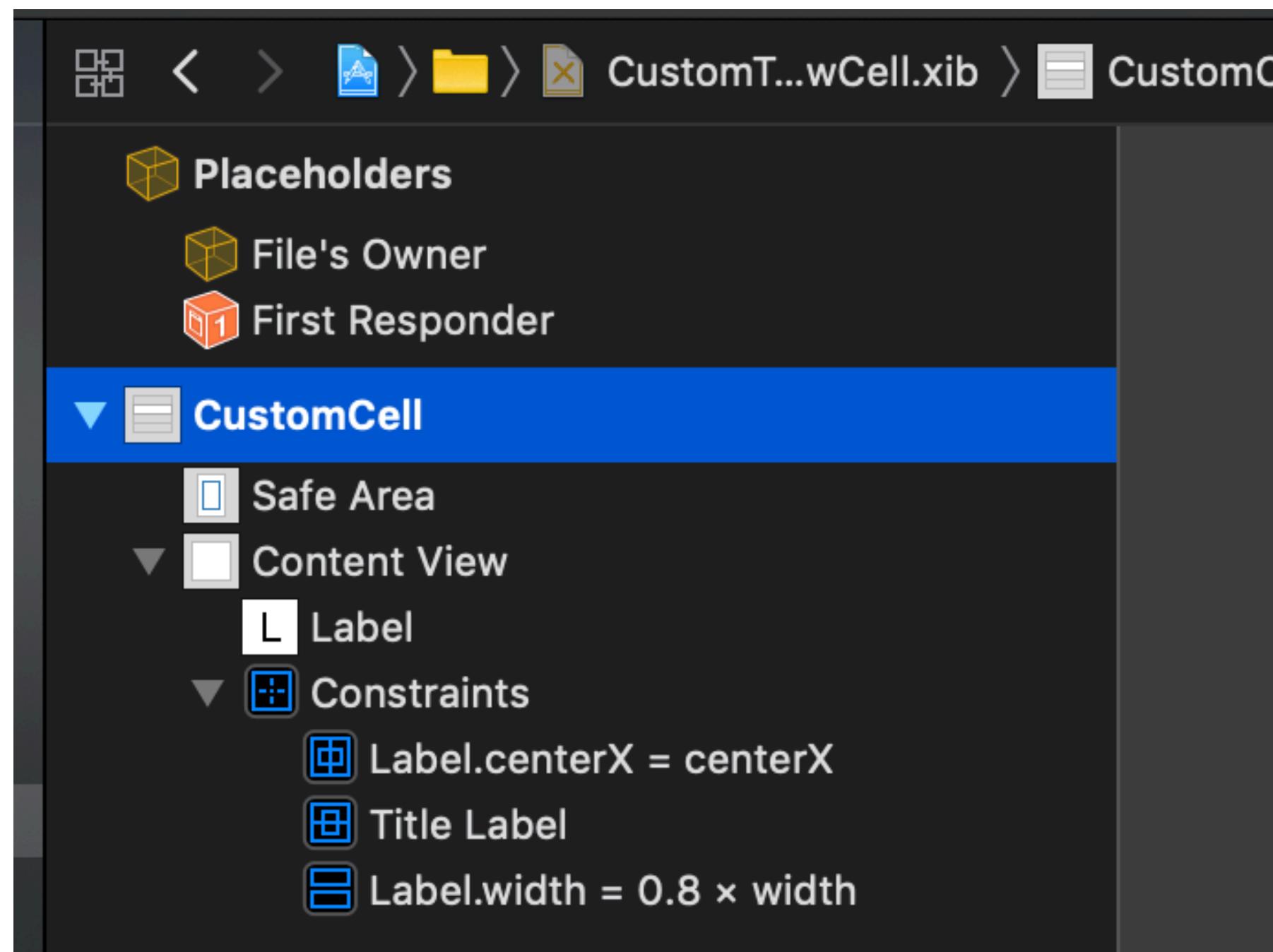
2-2. CellのSwiftファイルにOutletで接続



```
6 // Copyright © 2019
7 //
8
9 import UIKit
10
11 class CustomTableViewCell: UITableViewCell {
12
13     override func awakeFromNib() {
14         super.awakeFromNib()
15         // Initialization code
16     }
17
18     override func setSelected(_ selected: Bool, animated: Bool) {
19         super.setSelected(selected, animated: animated)
20     }
21 }
```

③CellのID設定

3-1. Xib上でCellを選択して、
Attributes Inspectorで「Identifier」を設定

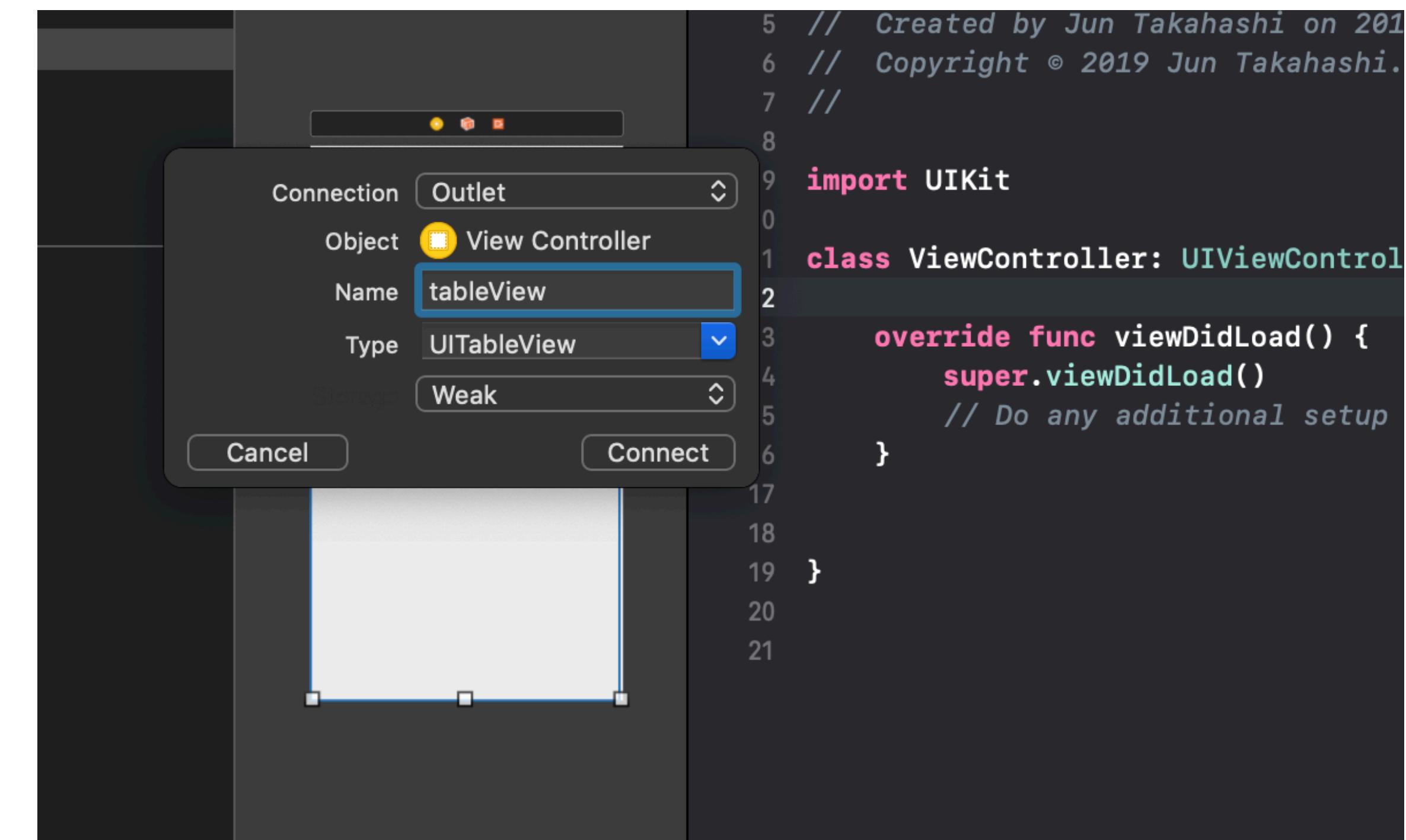
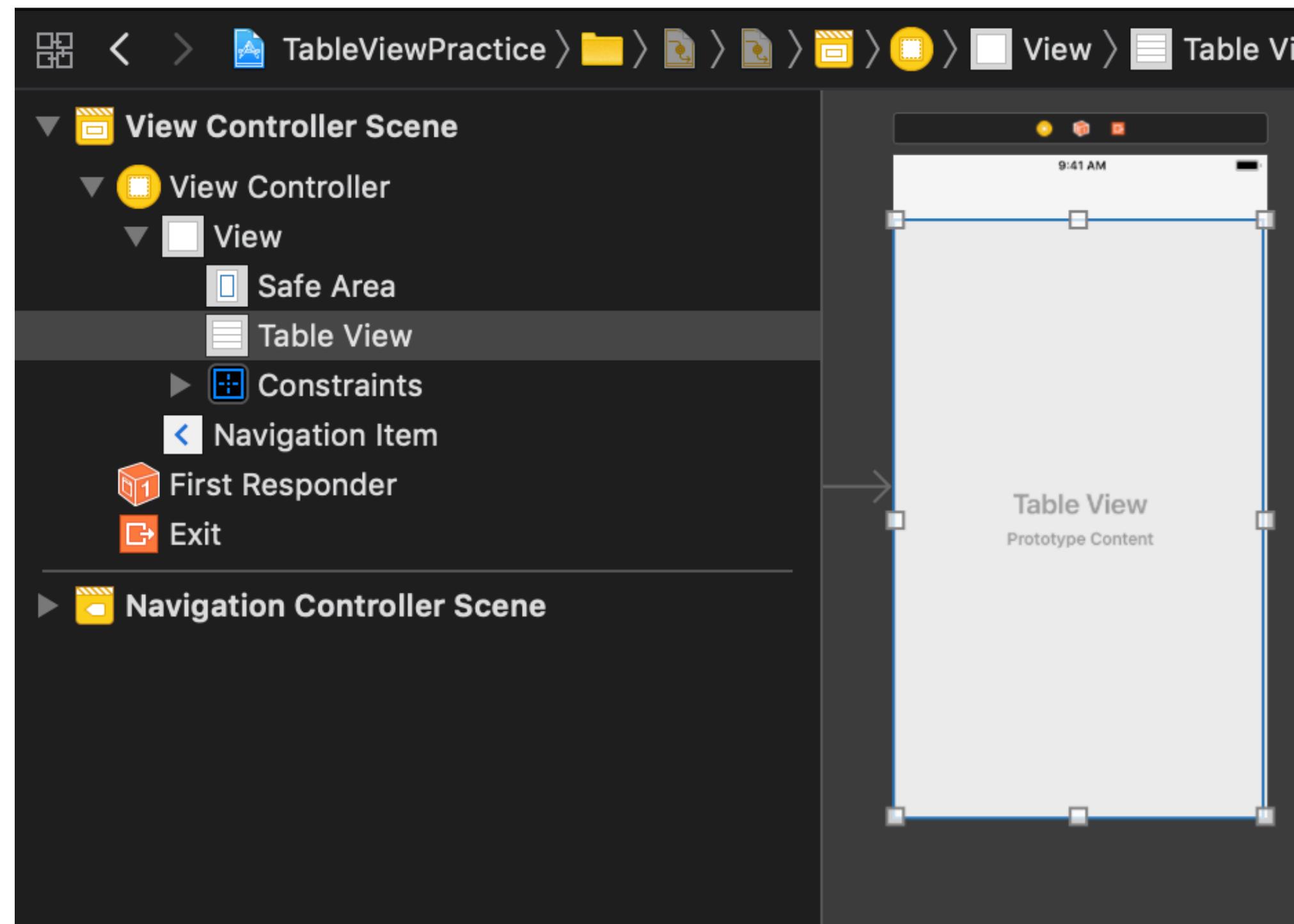


忘れがちなので注意！

④ VCにTableViewを配置

4-1. StoryBoard上でTableViewを配置

4-2. ViewControllerのSwiftファイルにOutletで接続



⑤ ViewControllerの設定

5-1. TableViewで作ったセルを利用できる状態にする

```
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.

    configureTableViewCell()
}

// TableViewCellを読み込む関数
func configureTableViewCell() {
    // TableViewCellのクラス名を指定してNibを作成
    let nib = UINib(nibName: "TableViewCell", bundle: nil)

    // Xibに設定したidentifier
    let cellID = "KameCell"

    // TableViewCellにcellのIdentifierを指定して登録
    tableView.register(nib, forCellReuseIdentifier: cellID)
}
```

コンパイル後のXibがNibなので、
Xib≡Nibと理解してOK

⑤ ViewControllerの設定

5-2. UITableViewの処理をVC側に委譲させる設定を行う

```
class ViewController:  
UIViewController, UITableViewDelegate, UITableViewDataSource
```

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    // Do any additional setup after loading the view.  
  
    // tableViewのUI処理を任せるのは自分だよ  
    tableView.delegate = self  
    // tableViewのデータ処理を任せるのは自分だよ  
    tableView.dataSource = self  
  
    configureTableViewCell()  
}
```

⑤ ViewControllerの設定

5-3. 必須で設定しなければならない

DataSourceメソッド「numberOfRowsInSection」で
返すセルの個数を決めてあげる

```
// tableViewのrowの数を返すDelegate
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return imageLists.count
}
```

⑤ ViewControllerの設定

5-4. 必須で設定しなければならないDataSourceメソッド「cellForRowAt」で返すセルの内容を決めてあげる

```
// tableViewのCellに表示する内容を返すDelegate(indexPathの個数だけ呼ばれるメソッド)
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    // Cellを呼び出すメソッド(ここは覚えてしまおう)
    let cell = tableView.dequeueReusableCell(withIdentifier: "KameCell", for: indexPath) as! TableViewCell

    // iconViewの設定
    cell.iconView.image = UIImage(named: imageLists[indexPath.row])
    // titleの設定
    cell.title.text = titleLists[indexPath.row]
    // descriptionの設定
    cell.descripiton.text = descripitonLists[indexPath.row]

    return cell
}
```

Cell内の情報をひとまとめにする

[File] - [New] - [File]から新しいファイル"KameInfo.swift"を作成

```
8
9 class KameInfo {
10     let iconName: String
11     let title: String
12     let description: String
13     init(iconName: String, title: String, description: String) {
14         self.iconName = iconName
15         self.title = title
16         self.description = description
17     }
18 }
19
```

その他のDelegateメソッド

```
// sectionの数を返すDelegate
func numberOfSections(in tableView: UITableView) -> Int {
    return 3
}

// sectionの数を返すメソッド
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    return "Section\(section)"
}

// tableViewのCellの高さを返すメソッド
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
    return 120.0
}

// tableViewのCellがタップされた時に呼ばれるメソッド
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    print("didSelectRowAt:", indexPath)
    let vc = DetailViewController()
    // タップされたKameInfoを遷移先のvcへ引き継ぐ
    vc.kameInfo = infoLists[indexPath.row]
    // 画面遷移
    navigationController?.pushViewController(vc, animated: true)
}
```

まだまだあります！

やってみよう



1. Navigation Controllerによるpush遷移
2. タップしたkamelInfoを引き継いで表示

遷移先のDetailViewController

```
9 import UIKit
10
11 class DetailViewController: UIViewController {
12
13     @IBOutlet weak var iconView: UIImageView!
14     @IBOutlet weak var titleLabel: UILabel!
15     @IBOutlet weak var descriptionLabel: UILabel!
16
17     var kameInfo: KameInfo?
18
19     override func viewDidLoad() {
20         super.viewDidLoad()
21
22         // Do any additional setup after loading the view.
23
24         // kameInfoをアンラップ
25         guard let kameInfo = kameInfo else { return }
26
27         // 遷移元から引き継がれた値をUIに反映
28         iconView.image = UIImage(named: kameInfo.iconName)
29         titleLabel.text = kameInfo.title
30         descriptionLabel.text = kameInfo.description
31     }
32 }
```

さいごに

次週予告！

1

ライブラリを使ってみよう

- ライブラリ管理ツール CocoaPodsの使い方

2

ToDoアプリを作ってみよう

- データの新規作成、読み込み、更新、削除(CRUD)
- 今後、このアプリを育てていきます。
(Firebaseを使ったデータ保存、画像ファイルの保存など)

【本日の課題】

お題 「TableViewを使ったアプリ」

(最低限の内容)

- xibを利用して画面をカスタムできる
- Cellをタップすると画面遷移を行う

*いろいろなDelegateの動きを試してみてください。

*余裕のある人はCollection Viewにチャレンジしてみてください。

(提出方法)

GitHubのリポジトリにpushし、そのURLを提出してください。

本日はお疲れ様でした。

