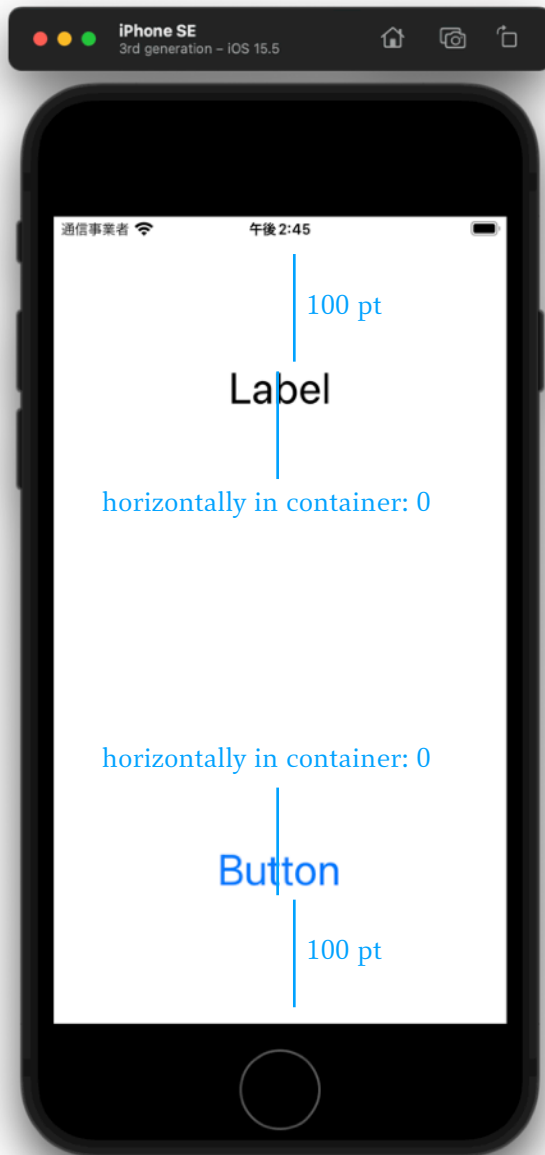


「iPhoneアプリ開発講座 はじめてのSwift」 UI & コードヒント集

2022/9/3 山田 泰司 <taiji@aihara.co.jp>

表題の書籍の p. 205 以降のすべてアプリを制作するにあたり、既に内容を理解した人が書籍を見ずに、そらで制作するための**最小限のヒント**をまとめておきます。私だけでなく、既に制作した方が復習したいときや制作時の最終確認などに有用かと思います。

§ 6.4 サンプルアプリ p. 220



アウトレット: Label → label

アクション: Button → buttonTapped

但し、本稿の Button はインスペクタペインのアトリビュートインスペクタにて、

その Button の Style: Plain → Default とし、フォントを System 36.0 としています。

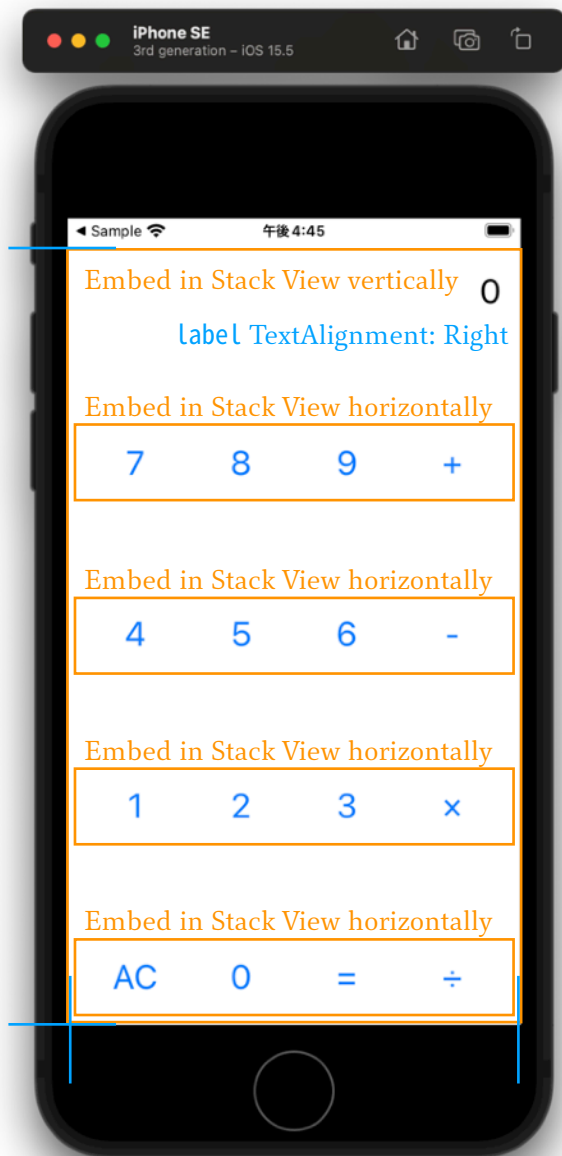
また、Label のフォントも System 36.0 としています。

🔖 書くコードのヒント

buttonTapped メソッド:

```
label.text = "Tapped!"
```

§ 7.1 電卓アプリ p. 246



vertical & horizontal Stacks

Alignment: Fill

Distribution: Fill Equally

vertical Stack constraints

bottom: 16

top: 16

leading: 16

trailing: 16

※ 書籍では 16 ではなくすべて 0

アウトレット: Label → label

アクション:

0～9 Button → numberButtonTapped

+ - × ÷ Button → operatorButtonTapped

= Button → equalButtonTapped

AC Button → allClearButtonTapped

但し、本稿の Button はインスペクタペインのアトリビュートインスペクタにて、

それら Button の Style: Plain → Default とし、フォントを System 28.0 としています。

また、Label のフォントも System 28.0 としています。

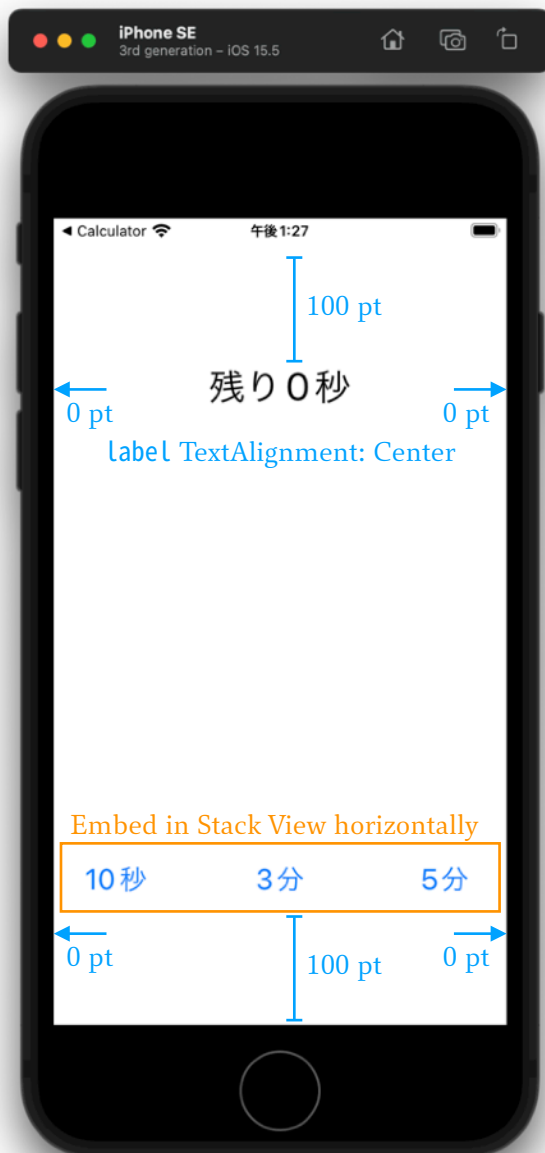
🔗 書くコードのヒント

numberButtonTapped メソッド:

```
value = Int(sender.currentTitle!) ?? 0
if currentOperator == .undefined {
    firstValue = firstValue * 10 + value
    label.text = "\(firstValue)"
}
else {
    secondValue = secondValue * 10 + value
    label.text = "\(secondValue)"
}
```

ここでは p. 260-261 のコード ②～③ を [nil 合体演算子](#)で簡潔に書いています。

§ 7.2 タイマーアプリ p. 266



horizontal Stack

Alignment: Fill

Distribution: Fill Equally

アウトレット: Label → label

アクション:

10秒 Button → tenSecButtonTapped

3分 Button → threeMinButtonTapped

5分 Button → fiveMinButtonTapped

但し、本稿の Button はインスペクタペインのアトリビュートインスペクタにて、

それらの Button の Style: Plain → Default とし、フォントを System 24.0 としています。

また、Label のフォントも System 32.0 としています。

🔗 書くコードのヒント

update メソッド:

```
currentSeconds -= 1
label.text = "残り\$(currentSeconds)秒"
if currentSeconds == 0 {
    timer?.invalidate()
    AudioServicesPlayAlertSound(1005)
}
```

start メソッド:

```
timer?.invalidate()
currentSeconds = seconds
label.text = "残り\$(currentSeconds)秒"
timer = Timer.scheduledTimer(timeInterval: 1.0,
    target: self,
    selector: #selector(ViewController.update),
    userInfo: nil, repeats: true)
```

ここでは p. 274 のコード ① 直前にタイマーを止めています。これがないとタイマーが複数発動、意図しない挙動となります。

§ 7.3 マップアプリ p.278



mapView constraints

bottom:	0 to View
top:	0 to View
leading:	0
trailing:	0

アウトレット: MKMapView → mapView

デリゲート: CLLocationManagerDelegate

デリゲートメソッド:

1. locationManagerDidChangeAuthorization(_ manager: CLLocationManager)
2. locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
3. locationManager(_ manager: CLLocationManager, didFailWithError error: Error)

プロジェクト — Info:

Privacy - Location When In Use Usage Description

🔗 書くコードのヒント

viewDidLoad メソッド:

```
manager.delegate = self
if manager.authorizationStatus != .authorizedWhenInUse {
    manager.requestWhenInUseAuthorization()
}
```

デリゲートメソッド 1:

```
if manager.authorizationStatus == .authorizedWhenInUse {
    manager.requestLocation()
}
```

デリゲートメソッド 2:

```
print("locations: \(locations)")
let current = locations[0]
let region = MKCoordinateRegion(center: current.coordinate,
    latitudinalMeters: 500, longitudinalMeters: 500)
mapView.setRegion(region, animated: true)
```

デリゲートメソッド 3:

```
print("error = \(error)")
```

ここでは p. 284~287 のコードにおいて、書籍から仕様が新しくなった書き方へ修正しています。

🔗 シミュレータでのアドバイス

Features メニュー — Location — Custom Location… にて以下の緯度経度を指定すれば「東京タワー」が拡大されます。

Latitude: 35.659138

Longitude: 139.745454

§ 7.4 図鑑アプリ p. 291



《ViewController》

アウトレット: TableView → tableView

デリゲート: UITableViewDataSource

デリゲートメソッド:

1. tableView(_ tableView: UITableView, **numberOfRowsInSection** section: Int)
2. tableView(_ tableView: UITableView, **cellForRowAt** indexPath: IndexPath)

Identifier: TableViewCell → NameCell

🔗 書くコードのヒント

```
struct AnimalInfo {
    var name: String
    var description: String
}
```

ViewController クラス:

```
let items = [
    AnimalInfo(name: "ゴリラ", description: "～"), ... ]
```

viewDidLoad:

```
tableView.dataSource = self
```

デリゲートメソッド 1:

```
return items.count
```

デリゲートメソッド 2:

```
let cell: UITableViewCell! =
    tableView.dequeueReusableCell(withIdentifier: "NameCell")
let item = items[indexPath.row]
var content = cell.defaultContentConfiguration()
content.text = item.name
cell.contentConfiguration = content
return cell
```

prepare メソッド:

```
if let selectedRow = tableView.indexPathForSelectedRow {
    let controller
        = segue.destination as! DetailViewController
    controller.info = items[selectedRow.row]
}
```

《DetailViewController》

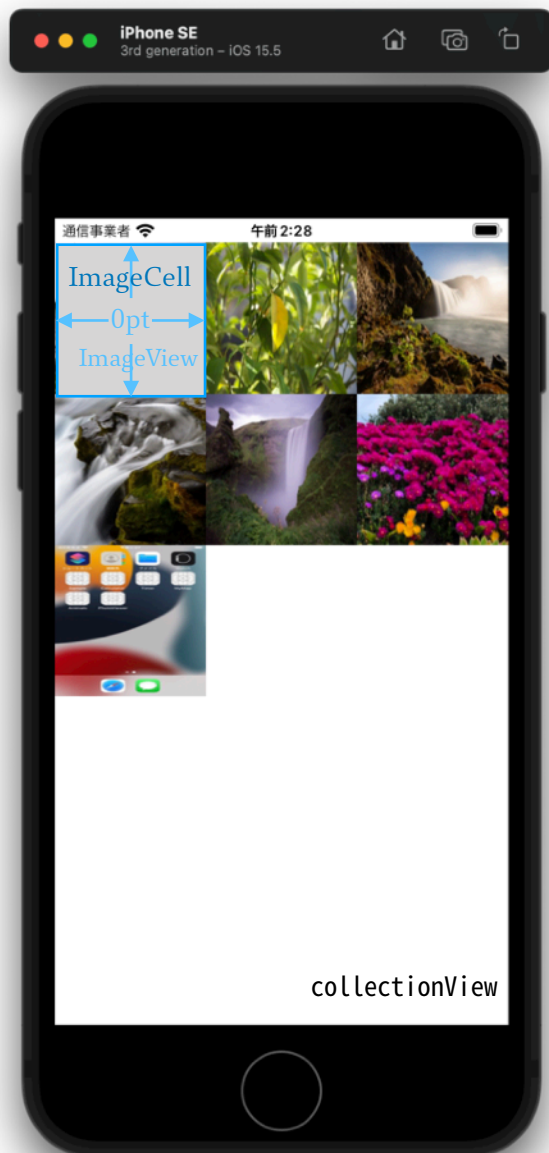
アウトレット: Label → label

🔗 書くコードのヒント

viewDidLoad メソッド:

```
navigationItem.title = info.name
label.text = info.description
```

§ 7.5 写真ビューアアプリ p. 313



collectionView constraints no margin

bottom:	0
top:	0
leading:	0
trailing:	0

ImageView constraints

bottom:	0
top:	0
leading:	0
trailing:	0

アウトレット: UICollectionView → collectionView

デリゲート: UICollectionViewDataSource,
UICollectionViewDelegateFlowLayout

デリゲートメソッド:

- collectionView(_ collectionView: UICollectionView, numberOfItemsInSection: Int)
- collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath)
- collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath)
- collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int)
- collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int)

Identifier: CollectionViewCell → ImageCell

Tag: ImageView → 1

プロジェクト — Info:

Privacy - Photo Library Usage Description

🔗 書くコードのヒント

ViewController クラス:

```
var photos: [PHAsset] = []
let manager = PHImageManager()
```

loadPhotos メソッド:

```
let result = PHAsset.fetchAssets(with: .image,
options: nil)
let indexSet = IndexSet(integersIn: 0..

```

viewDidLoad メソッド:

```
collectionView.delegate = self
collectionView.dataSource = self
PHPhotoLibrary.requestAuthorization({ status in
if status == .authorized {
self.loadPhotos()
}
})
```

デリゲートメソッド: 次ページに続く 🔗

🔗 前ページの続き

デリゲートメソッド 1 (`numberOfItemsInSection`):

```
//return 0
return photos.count
```

デリゲートメソッド 2 (`cellForItemAt`):

```
//return UICollectionViewCell()
let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "ImageCell",
                                             for: indexPath)

let asset = photos[indexPath.item]
let width = collectionView.bounds.width / 3
manager.requestImage(for: asset, targetSize: CGSize(width: width, height: width),
contentMode: .aspectFill, options: nil, resultHandler: { result, info in
    if let image = result {
        let imageView = cell.viewWithTag(1) as! UIImageView
        imageView.image = image
    }
})
return cell
```

デリゲートメソッド 3 (`sizeForItemAt`):

```
let width = collectionView.bounds.size.width / 3
return CGSize(width: width, height: width)
```

デリゲートメソッド 4 (`minimumInteritemSpacingForSectionAt`):

```
return 0.0
```

デリゲートメソッド 5 (`minimumLineSpacingForSectionAt`):

```
return 0.0
```

ちなみに、表題の書籍と全く同一の書き方ではない箇所があります（作用は同一）が、わかりやすさの為に意図して書き方を変えており、書籍と違いを見比べてみるとよいでしょう。