@ViewLoadingプロパティラッパの紹介と自前で実装する方法

宇佐見公輔 / 株式会社ゆめみ

2023-08-10

自己紹介

- 宇佐見公輔（うさみこうすけ）
- 株式会社ゆめみ / iOSテックリード
- 大阪在住
- iOSDC Japan 2023
  - パンフレット記事を執筆しました
  - 現地参加予定

今日の内容

- `@ViewLoading` プロパティラッパの紹介
- `@ViewLoading` を自前で実装する方法

`@ViewLoading` プロパティラッパの紹介

`@ViewLoading` とは

- iOS 16.4で追加された
  - 2023年3月リリース
  - マイナーアップデートでのSDKの機能追加はめずらしい
- `UIViewController` のプロパティが扱いやすくなる

`UIViewController` のプロパティ

```
class DateViewController: UIViewController {
    private var dateLabel: UILabel! // Optional型

    override func viewDidLoad() {
        super.viewDidLoad()
        let label = UILabel(frame: self.view.bounds)
        self.view.addSubview(label)
        self.dateLabel = label
    }
}
```

（このコード例はAppleのドキュメントから引用）

`UIViewController` のプロパティ

```
class DateViewController: UIViewController {
    private var dateLabel: UILabel! // Optional型
```

- 一度設定したら `nil` にならないのでOptional型は冗長に感じる
- しかし、非Optional型にする場合は初期化時に値の設定が必要
- `viewDidLoad()` 時に値を設定するにはOptional型にする必要がある

@ViewLoading プロパティラッパ

```
class DateViewController: UIViewController {
    @ViewLoading private var dateLabel: UILabel // 非Optional型

    override func viewDidLoad() {
        super.viewDidLoad()
        let label = UILabel(frame: self.view.bounds)
        self.view.addSubview(label)
        self.dateLabel = label
    }
}
```

`@ViewLoading` プロパティラッパ

```
class DateViewController: UIViewController {
    @ViewLoading private var dateLabel: UILabel // 非Optional型
```

- 非Optional型で宣言できる
- 内部的にはOptional型で値を持っている
- `get` アクセス時に内部の値をunwrapして返してくれる
  - → アクセス時にまだ値が設定されていなかったら？

`@ViewLoading` プロパティラッパ

- `get` アクセス時にViewのロードを行ってくれる
- `get` アクセス時
  - → `loadView()` が実行される
  - → `viewDidLoad()` が実行される
  - → プロパティの値が返される
- このため、`viewDidLoad()` でプロパティの値を設定すれば良い

注意点

- `viewDidLoad()` で値を設定し忘れると実行時エラーになる
  - `nil` のunwrapになってしまうため
- `@ViewLoading` プロパティラッパを指定したプロパティは、必ず `viewDidLoad()` で値を設定すること

利用例

```
class DateViewController: UIViewController {
    var date: Date? {
        didSet {
            guard let date else { return }
            let dateString = self.dateFormatter.string(from: date)
            self.dateLabel.text = dateString
        }
    }
}

let dateViewController = DateViewController()
dateViewController.date = Date()
```

利用例

```
var date: Date? {
    didSet {
        self.dateLabel.text = dateString
    }
}
```

- `date` の `didSet` の中で `dateLabel` にアクセスしている
  - この時点で `dateLabel` の値が設定されている必要がある

利用例

```
let dateViewController = DateViewController()
dateViewController.date = Date()
```

- `date` へのアクセス時点ではViewのロードが行われていない
- このため `dateLabel` について
  - `@ViewLoading` を使っていない場合はエラーになる
  - `@ViewLoading` を使っている場合は正常に動作する

`@ViewLoading` を自前で実装する方法

`@ViewLoading` を古いOSでも使いたい

- 便利な機能だがiOS 16.4以降でしか使えない
- それ以前のバージョンで使うために自前で実装することを考える

`@ViewLoading` を自前で実装する

- プロパティラッパはSwiftの機能であり自作できる
- しかし `@ViewLoading` の実現は通常の方法ではどうも難しい
  - とくに `UIViewController` のメソッドを呼ぶ手段がわからない
- リファレンス実装を作っている人がいたので参考にする
  - https://github.com/danielpunkass/MagicLoading

プロパティラッパの通常の実装方法

```
@propertyWrapper
struct TwelveOrLess {
    private var number = 0
    var wrappedValue: Int {
        get { return number }
        set { number = min(newValue, 12) }
    }
}
```

- `wrappedValue` を実装する
- このプロパティを含む構造体やクラスからは独立している

プロパティラッパの第二の実装方法

```
@propertyWrapper
struct MagicViewLoading<Value> {
    static subscript<T>(
        _enclosingInstance instance: T,
        wrapped wrappedKeyPath: ReferenceWritableKeyPath<T, Value>,
        storage storageKeyPath: ReferenceWritableKeyPath<T, Self>
    ) -> Value  {
        ...
    }
}
```

プロパティラッパの第二の実装方法

- この実装方法なら、プロパティを含むクラスにアクセスできる
- この実装方法はSwiftのドキュメントには記載されていない
- プロポーザルには記載がある
  - swift-evolution SE-0258 Property Wrappers
- Swift by Sundellで紹介されている
  - Accessing a Swift property wrapper's enclosing instance | Swift by Sundell

`@ViewLoading` の実装

```
static subscript<T: UIViewController>(
    _enclosingInstance instance: T,
    wrapped wrappedKeyPath: ReferenceWritableKeyPath<T, Value>,
    storage storageKeyPath: ReferenceWritableKeyPath<T, Self>
) -> Value {
    get {
        instance.loadViewIfNeeded()
        return instance[keyPath: storageKeyPath].stored!
    }
    set {
        instance[keyPath: storageKeyPath].stored = newValue
    }
}
```

まとめ

- iOS 16.4で `@ViewLoading` という便利機能が追加された
- それ以前のバージョンでも同様の機能の実現が可能
  - ただしSwiftのドキュメントに記載されていない方法なので注意