iOS Seminar 3

Wafflestudio 2024

과제 2 공통 피드백

Enumeration (관련 문서)

- path, parameter 등... 매 Request마다 지정해서 보내야 하는 값
  - 매번 새 변수로 선언하게 되면, `let parameters: Parameters = [...]`의 반복
- 여러 case에서 공통되는 Type의 Variable을 사용해야 할 때
  - Enumeration+Switch문의 활용 : 코드 반복을 줄일 수 있다!

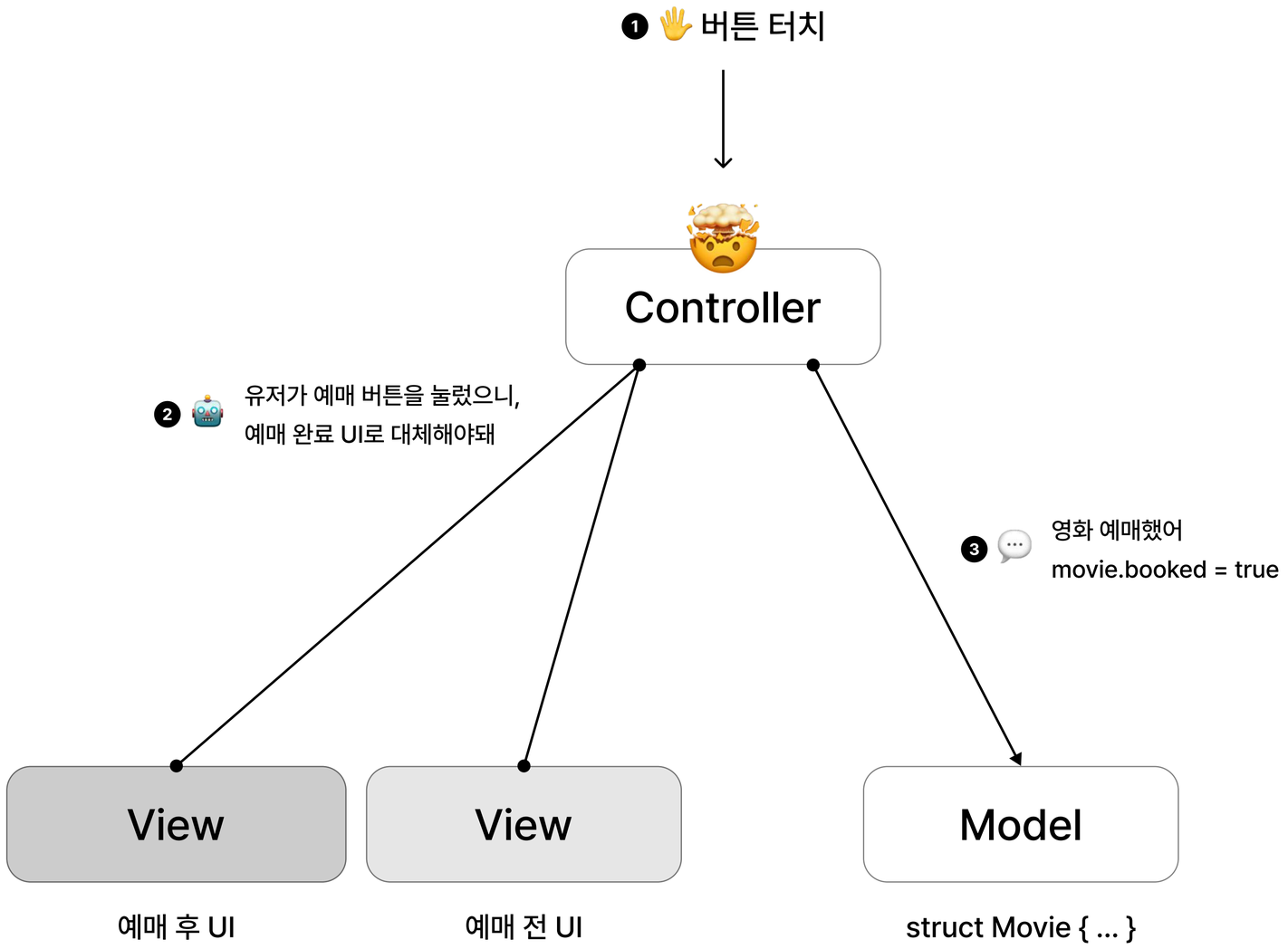
Design Pattern

Design Pattern

- MVC
- MVVM
- 그 외 MVP 등...

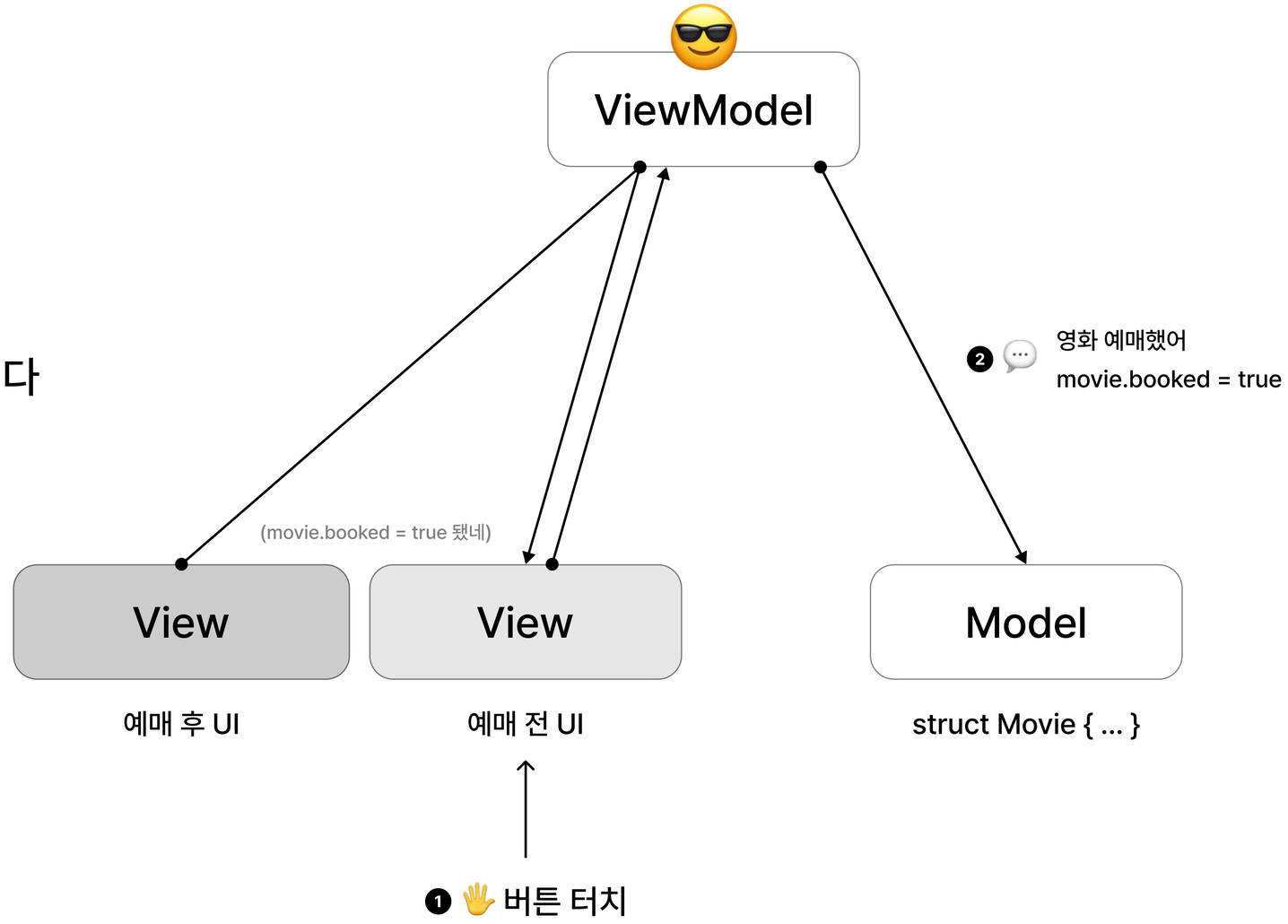
MVC

- Model - View - Controller
- 이제까지 진행했던 과제와 유사한 패턴
- Massive - View - Controller
  - 규모가 큰 프로젝트일수록 잘...



MVVM

- Model - View - ViewModel
- View와 Model 간의 결합도를 없앰
  - ViewModel이 바뀌면 View도 바뀐다



SwiftUI

SwiftUI

- WWDC 2019에서 발표한 프레임워크
- Declarative syntax
  - (UIKit은 Imperative)
- Data Driven
- HStack, VStack, ZStack, Spacer 등을 이용

```swift
import SwiftUI

struct MainView: View {

    @State private var id: String = ""
    @State private var password: String = ""
    @State private var pushToAfterLoginView: Bool = false

    var body: some View {
        VStack {
            VStack(alignment: .leading, spacing: 0) {
                Text("iOS Seminar")
                    .font(.system(size: 28, weight: .semibold))

                Spacer().frame(height: 48)

                VStack(spacing: 32) {
                    TextField("아이디", text: $id, prompt: Text("아이디(8자 이하)"))
                        .font(.system(size: 17))
                        .textInputAutocapitalization(.never)

                    TextField("비밀번호", text: $password, prompt: Text("비밀번호"))
                        .font(.system(size: 17))
                        .textContentType(.password)
                        .textInputAutocapitalization(.never)
                }
            }
            .padding(.horizontal, 24)

            /// ...
```

UIKit보다 좋은 점?

- 과제1의 TodoTableViewCell을 SwiftUI로 구현한다면...
- ex) 좌우 간격 24, 상하 간격 16을 설정하고자 했을 때
  - UIKit

```
98      private func setLayoutConstraint() {
99          addSubview(labelStackView)
100         labelStackView.translatesAutoresizingMaskIntoConstraints = false
101         NSLayoutConstraint.activate([
102             labelStackView.leadingAnchor.constraint(equalTo: checkImageView.trailingAnchor, constant: 24),
103             labelStackView.trailingAnchor.constraint(equalTo: safeAreaLayoutGuide.trailingAnchor, constant: -24),
104             labelStackView.topAnchor.constraint(equalTo: safeAreaLayoutGuide.topAnchor, constant: 16),
105             labelStackView.bottomAnchor.constraint(equalTo: safeAreaLayoutGuide.bottomAnchor, constant: -16),
106         ])
107     }
```

  - SwiftUI

```
42          .padding(.horizontal, 24)
43          .padding(.vertical, 16)
```

그럼에도 UIKit을 배우는 이유

- 복잡한 custom UI를 구현하고자 하는 경우
  - UIKit + SwiftUI 혼용하는 프로젝트도 다수 존재
- iOS 15까지 pure SwiftUI로는 Alert에 TextField를 넣을 수 없었다...

SwiftUI Life cycle

UIKit → SwiftUI Life cycle (공식문서)

- 기존 UIKit을 사용했을 때 앱의 entry point는 `@main` annotation이 붙어있던 AppDelegate
  - `@main`: 특정 structure, class, 혹은 enumeration이 top-level entry point임을 나타낸다
  - C의 `int main(int argc, char *argv[])`같은 느낌
- SwiftUI에서는 `<MyAppName>App.swift`
- 관련 문서 (SwiftUI Tutorial)
- 물론 기존의

```swift
import SwiftUI

@main
struct Seminar3DemoApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

SwiftUI+Data Managing

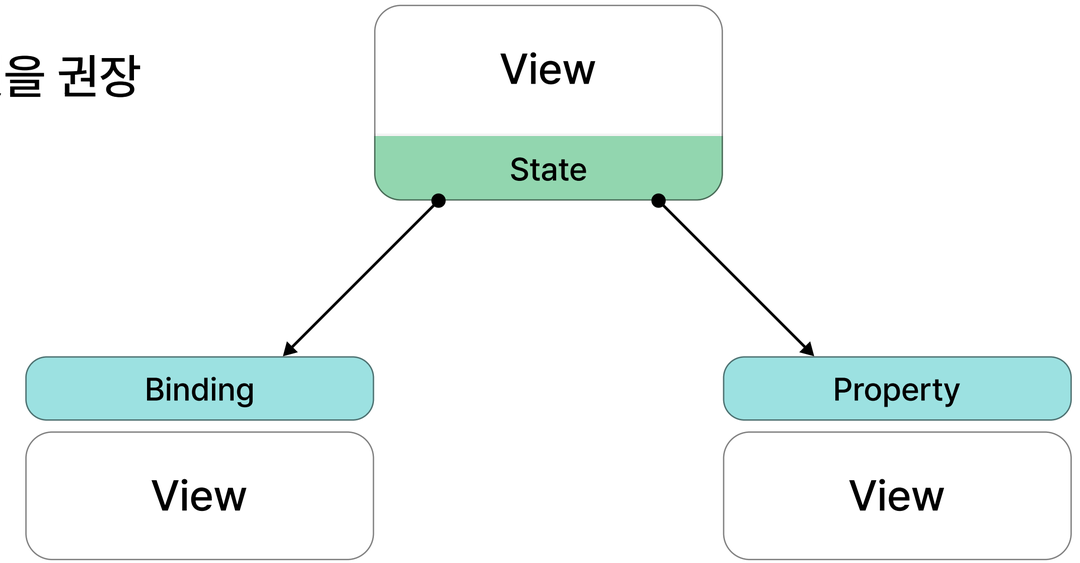
SwiftUI가 View를 업데이트하는 방식

- 관련 문서
- SwiftUI는 Data Driven
  a. View state 공유
  b. Model data의 변경사항 관찰 (이건 다음 세미나에서)
- View에 의존성이 있는 데이터가 업데이트되면, 그 데이터에 연관된 인터페이스만 자동으로 업데이트

View State Sharing

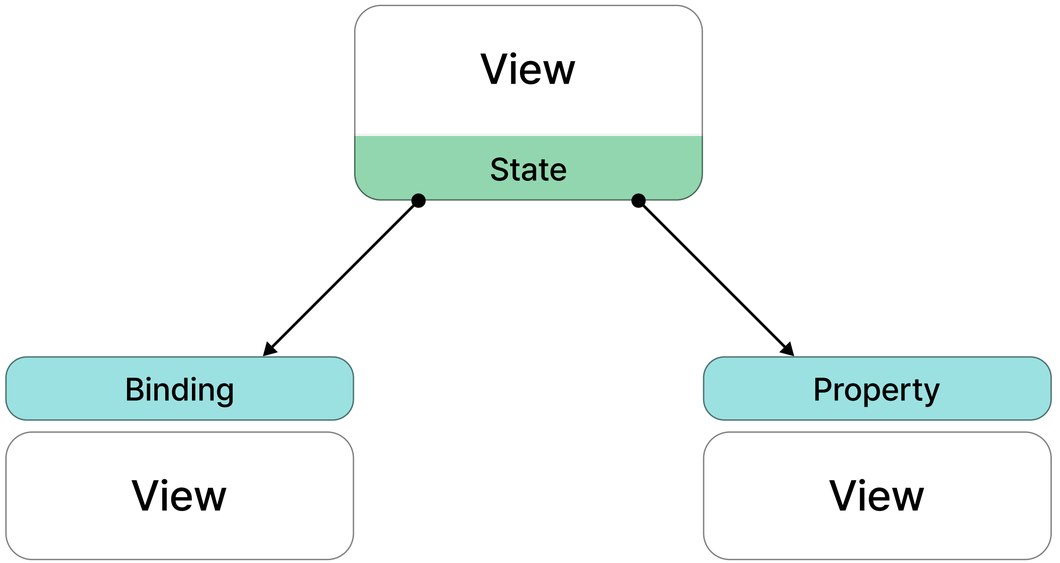
- 관련문서
- Store data as state in the least common ancestor of the views that need the data to establish a single source of truth that’s shared across views.
- 다만 이러한 state는 persistent한 데이터보다는,

  User Interface에만 영향을 미치는 데이터에만 적용할 것을 권장



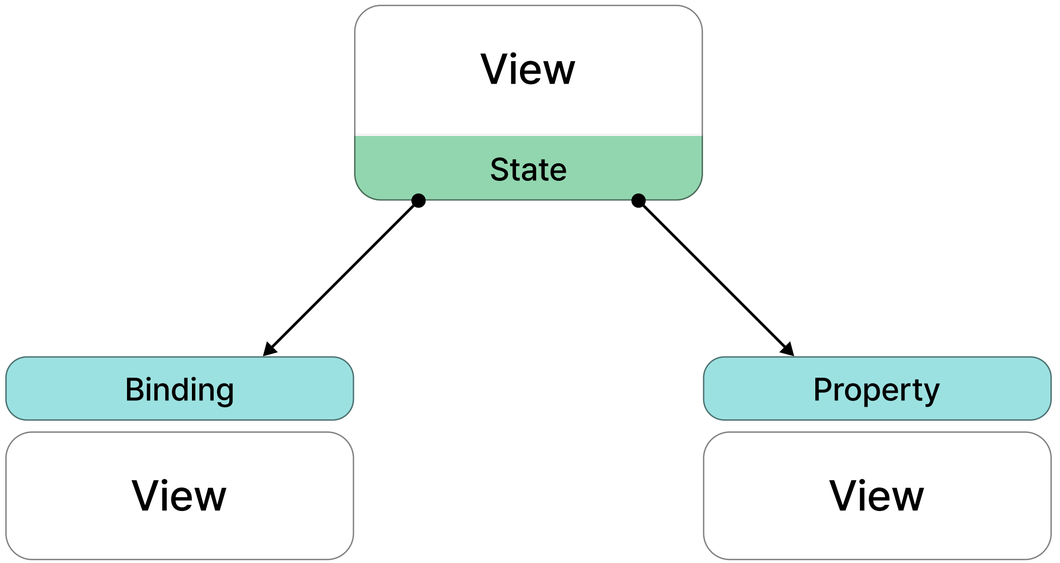
@State (공식문서)

- User Interface state를 지역적으로 관리하는 경우
  - ex) 화면의 counter 값을 1씩 올리기
- 하위 View에게 read-only로 공유하거나, binding을 이용해 read-write로 공유
  - binding으로 공유하는 경우 `$`
- 어느 스레드에서든 안전하게 값 변경 가능



@Binding (공식문서)

- 상위 View의 state를 받아 사용하면서도, 현재 View(하위 View)에서의 변경사항이 상위 View에 영향을 미치도록 하고 싶은 경우
- ex) 상세 페이지에서 Toggle을 on/off는 경우에 따라 메인 페이지를 다르게 보여주기



과제 3 안내

SwiftUI를 이용한 과제1 리메이크

- 11/2(토) 업로드 예정, 11/16(토) 오전 8시 마감
- 과제 목표
  - UIKit → SwiftUI migration을 통해 둘의 차이 이해
- 관련 키워드
  - SwiftUI
  - UserDefaults