



iOS Seminar 1

한상현

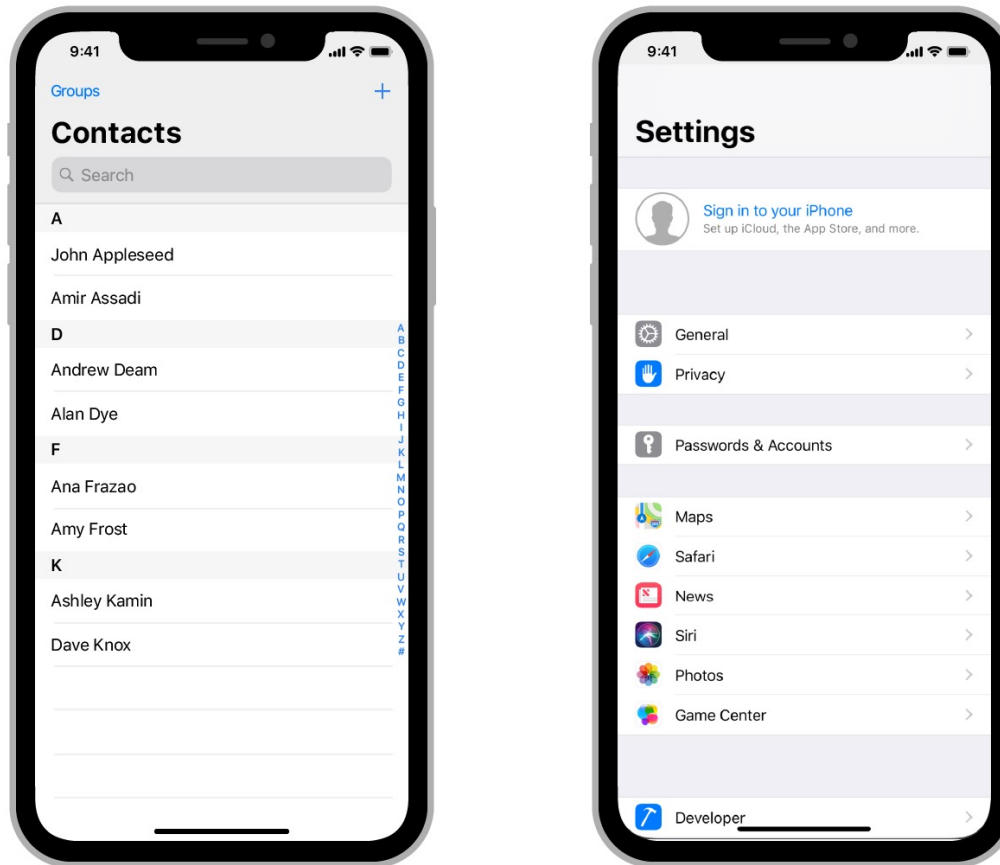
세미나장 소개 - 한상현

The logo for UPbit, featuring the text "UPbit" in a bold, blue, italicized sans-serif font. The "U" and "P" are connected, and the "i" has a dot. The "b" is also connected to the "i".

오늘 배울 내용

- UITableView : 열이 하나인 표 형태의 View
- Protocol : 우리가 정한 규약, 특정 객체에게 정의할 메소드를 미리 지정해준다
- Delegate 패턴 : 내가 해야할 일을 대리자에게 맡기는 코드 구현 방식
- UIStackView : 다르게 생긴 여러 View들을 쌓아 올린 View
- UserDefaults : 가벼운 Local 저장소
- Codable : JSON <-> Custom Object 변환을 위한 애플의 솔루션

UITableView



UITableViewCell

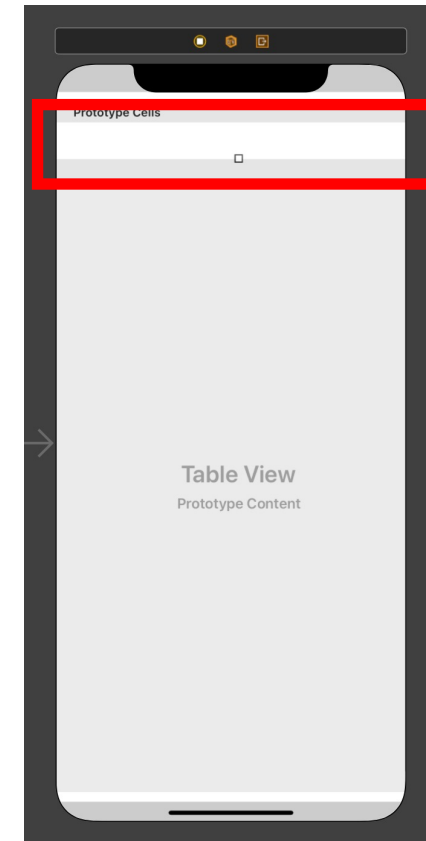
Class

UITableViewCell

The visual representation of a single row in a table view.

Declaration

```
@MainActor class UITableViewCell : UIView
```



Protocol 이란?

- 약속, 규약 -> 우리가 구현하기로 약속한 메소드들의 집합
- 이게 왜 필요한가?: 같은 기능을 하는 다른 객체들에 누가 봐도 알아볼 수 있는 표식을 심어준다고 생각하면 편함 (OOP의 다형성 개념)

```
protocol TestProtocol {  
  
    func waffleiOS(input: String) -> Bool  
  
    func waffleSeminar(type: String)  
}
```

```
class TestClass: TestProtocol {
```



Type 'TestClass' does not conform to protocol 'TestProtocol'

```
}
```

Delegate 란?

- 어떤 객체에게 필요한 작업을 다른 객체가 위임받아 대신 하는 것!
- Delegate 사전적 의미 : 위임하다
- + 상속의 개념을 꼭 알고 계셔야 해요! (OOP 개념)

```
extension MainViewController: UITableViewDelegate {  
  
}  
  
extension MainViewController: UITableViewDataSource {  
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
        code  
    }  
  
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
        code  
    }  
}
```

TableView 구현을 위한 Delegate 구현

- TableView의 셀이 몇 줄인지?
- 각 셀에는 어떤 데이터가 들어가야 하는지?
- 셀이 선택되면 어떤 작업을 이어서 진행할 것인지?

-> 이 모든 작업을 TableView를 가지고 있는 ViewController가
대리하여 진행한다!

-> ViewController가 Delegate를 상속받아야 함

TableView 구현을 위한 Delegate 구현

```
class MainViewController: UIViewController {  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
    }  
}  
  
extension MainViewController: UITableViewDelegate {  
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  
        code  
    }  
}  
  
extension MainViewController: UITableViewDataSource {  
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
        code  
    }  
  
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
        code  
    }  
}
```

Xcode로 고고!



UIStackView

- Stack : 쌓는 것
→ UIStackView : 쌓는 뷰
- TableView는 비슷한 모양의 셀을 여러 개 사용해야 할 때
- StackView는 표 형태로 여러 다른 형태의 뷰를 쌓아서 사용해야 할때!
: 뷰의 개수가 정해져있고, 모양이 다를 때 사용하는 것이 유리!

Xcode로 고고!



UserDefaults

- 기기의 로컬 저장소에 데이터를 저장하는 방법은 총 네 가지!
1. UserDefaults : 가벼운 정보들
 2. CoreData : DB를 구축해야 하는 양이 많은 데이터
 3. FileManager : 이미지와 같이 파일의 형태로 변환해서 저장해야하는 정보들
 4. KeyChain : 비밀번호와 같이 암호화가 필요한 정보들 + Apple ID 연동 가능

UserDefaults의 아주 간단한 사용법

```
func saveData() {  
    UserDefaults.standard.setValue("Data", forKey: "Data")  
}  
  
func getData() {  
    let _ = UserDefaults.standard.data(forKey: "Data")  
    let _ = UserDefaults.standard.string(forKey: "Data")  
}
```

우리는 Custom 클래스 정의해서 쓸건데?

- UserDefaults는 특정 type만 저장할 수 있게 되어있음!
- (String, Bool, Int, Data 등등)
- 우리가 만든 Class, Struct를 저장하려면???
- > 객체에 **Codable** Protocol 적용 -> **Data**로 변환하여 저장

Codable이란?

- 말 그대로 Code + able : 시스템이 key - value를 판단할 수 있도록 구현
- Decodable, Encodable : 두 개의 Protocol에 맞춰 구현해야 한다

Type Alias

Codable

A type that can convert itself into and out of an external representation.

Declaration

```
typealias Codable = Decodable & Encodable
```

Discussion

Codable is a type alias for the Encodable and Decodable protocols. When you use Codable as a type or a generic constraint, it matches any type that conforms to both protocols.

Codable 사용법

- Protocol만 적용해주면 간단!
- 다만 String, Bool, Int 이런 자료형이 아니라 Custom 된 객체 혹은 추가적인 작업이 필요한 경우에는 protocol에 맞춰 작업을 해줘야 함
- Decodable : init(from: Decoder) -> Decoder가 객체 init 하는 함수
- Encodable : encode(to: Encoder) -> Encoder가 객체 encoding하는 함수

```
class Task: Codable {  
    let description: String  
    let done: Bool  
}
```

Assignment 2 - TodoList 구현

- 해야 할 일을 등록하고, 완료 여부를 기록할 수 있는 기본적인 TodoList를 구현
 - 앱을 켜다 키더라도 기존에 등록한 Todo Task들이 그대로 노출되어야 함
 - NavigationBar Right Button에 버튼을 넣고 그 버튼을 탭할 경우 Task 추가 뷰로 이동하고, 그 뷰에서 입력 완료 버튼을 누르면 다시 List 뷰로 돌아오도록 구현 필요. 그리고 즉시 새로 추가된 Task가 List에 포함되어야 함.
 - 각 Task에 해당하는 셀에는 Task의 이름과 완료 여부가 표시되어야 함.
 - Task를 완료하기 위해서는 셀 우측에 있는 완료 버튼을 눌러야 함
-> Done 되었을 경우 이름이 무언가 바뀌어야 함 (**이름**, 완료)
- 추가 기능 : 수정 + 삭제 기능 (어떻게 하면 좋을지는 구글링해서 찾아보시길...)

Assignment 2 - TodoList

- 개발에 필요한 지식
 1. UITableView + UITableViewCell
 2. UserDefaults or FileManager
 3. Codable
 4. UINavigationController

<추가 기능 개발 시 참고>

- 애플이 TableViewCell에 쓰라고 준 삭제 기능이 있다던데?
(왼쪽으로 쓱 밀면 나오는 그거?!)
- 꾸욱 누르면 삭제하시겠습니까?를 뜨게 할 수도 있지 않을까? (UIGestureRecognizer)

혼자 공부해보면 좋은 내용

- 공식 문서 :
https://developer.apple.com/documentation/uikit/views_and_controls/table_views
- RxDataSource (나중에 가르쳐 드릴 예정) :
<https://github.com/RxSwiftCommunity/RxDataSources>
- Rx로 TableView 만들기 :
<https://eunjin3786.tistory.com/29>
- TableView 개념 익히기 : <https://zeddios.tistory.com/55?category=682195>
- Codable 심화 :
<https://minsone.github.io/programming/swift-codable-and-exceptions-extension>