
# iOS Seminar 2

한상현

# 오늘 배울 내용

- 글로벌한 데이터 관리 방법

  : Usecase, Manager (Singleton Pattern)
- Git 사용에 대하여
- Network / HTTP
- URLSession / Alamofire
- Cocoapods / Swift Package Manager -> 3rd party 관리
- OpenAPI
- RxSwift + MVVM / + UITableView (Optional)

# Singleton Pattern

- Class 내부에 static var shared = ClassName()의 방식으로 객체를 만들어 Class 자체의 하나의 instance를 자동 생성하는 방식
- 클래스 내에 데이터를 저장하고 관리하는 로직을 추가할 수 있기 때문에 단순 글로벌 변수에 비해 더 체계적으로 method를 통한 데이터 관리가 가능

- 다만 여러 곳에서 동시에 접근이 가능하다는 건 set과 get에 대한 타이밍 이슈가 항상 발생할 수 있음을 의미
  -> 간단할땐 괜찮은데 복잡하면 문제가 될수도? (온갖 View들에서 다 싱글톤 내의 데이터를 수정하는 경우)

# Singleton 예제

```swift
struct TestModel {
    let content: String
}

class TestManager {
    static let shared = TestManager()
    var tests: [TestModel] = []

    func savetests() {

    }

    func updateTests() {

    }

    func loadTests() -> [TestModel] {
        return self.tests
    }
}
```

# Usecase

- 여기저기서 쓰이는 데이터의 관리
  : 앱 전반의 로직을 관장하는 역할
  : Domain(도메인) 이라는 이름으로 주로 부름

- 각각의 기능에 맞춰서 필요한 데이터만 가져다 쓰고, 저장 가능하게 구현
  : Dependency Injection (의존성 주입)의 개념에서 그 때 그 때 전달하면서 사용하는 개념.

- 싱글톤만 아닐 뿐이지 기능 자체는 차이가 없음

-> 얘는 기회가 되면 일요일 코모에서 설명드림

# Git 사용에 대하여

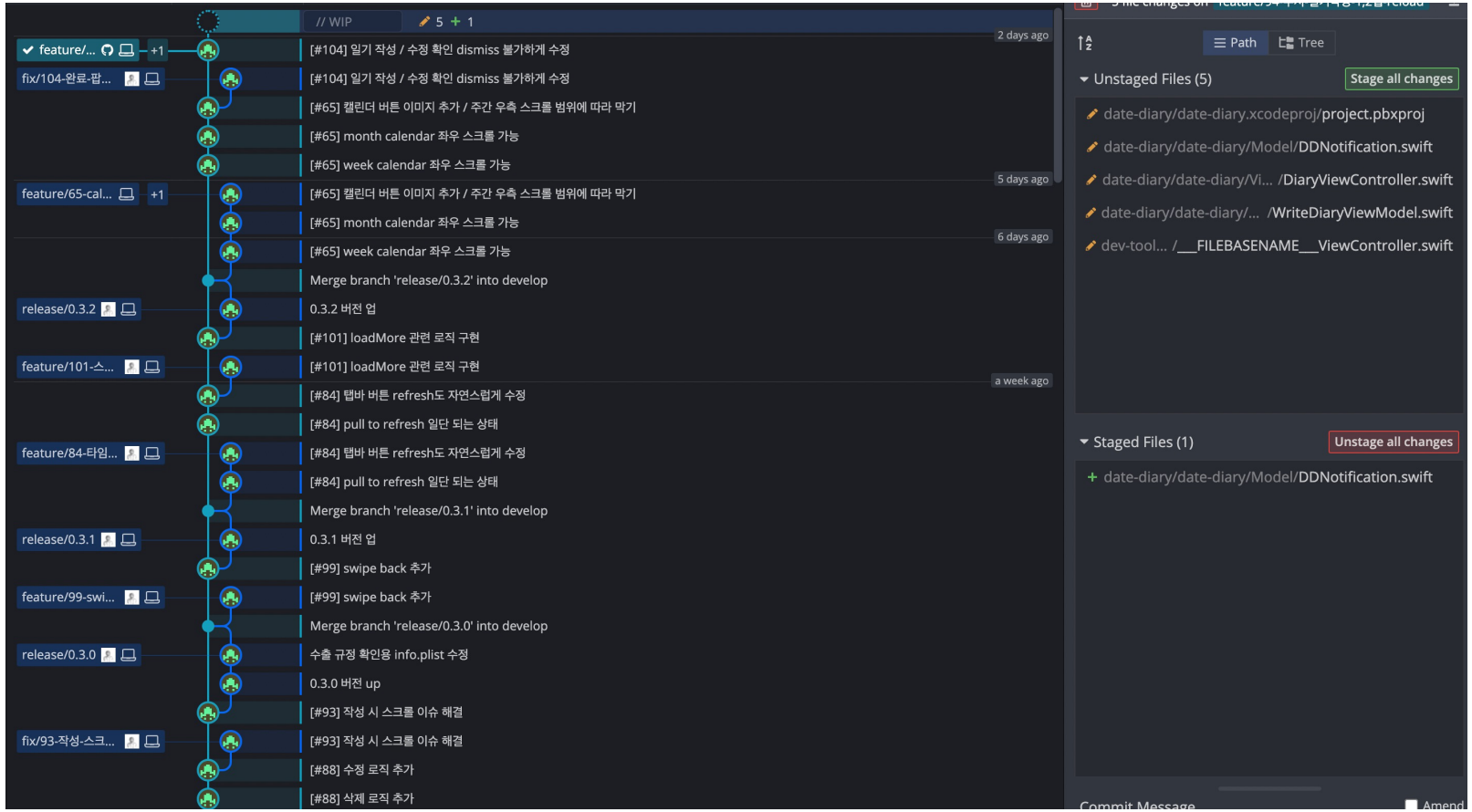
- Git 사용이 생소한 건 당연한 거지만 그렇다고 어려우니까 공부를 안 하시면 안 됩니다!!
- 지금은 과제니까 적당히 브랜치 꼬여도 삭제하고 다시 하면 됨
  : 와플 프로젝트 들어가서는… 불가능
- 보통 깃을 어려워 하는 이유가 CLI가 직관적이지 않아서
  -> GUI 툴을 사용합시다

# GitKraken / Fork

- Fork는 무료인 대신 기능이 좀 적고 UI가 불편한 부분이 존재
- GitKraken은 유료이지만 대신 education license로 사용이 가능

  (Github Educational Pack에 가입해두세요!! 그럼 깃헙 계정으로 연동 가능)

# 저는 GitKraken을 씁니다



# 회사나 단체마다 Git 사용법은 다름

- Git의 기본적인 커밋, 머지, 풀리퀘스트 등의 기능은 모든 곳에서 사용
- 다만 이를 어떤 전략으로 관리할지는 회사마다 다르고, 개발자들마다 다를 수 있음

- 깃 브랜치 전략에 대한 내용을 읽어보시는 것도 나쁘지 않음 (예습처럼)
  https://velog.io/@kw2577/Git-branch-%EC%A0%84%EB%9E%B5

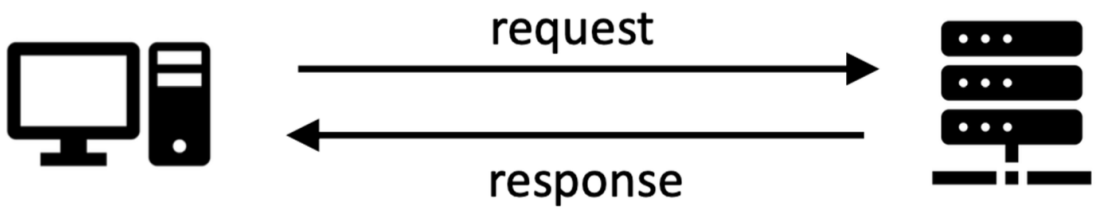
# Network

HyperText Transfer Protocol

서버와 클라이언트가 데이터를 주고받기 위해 정의한 규칙

Http에 의해 서버는 클라이언트가 요청할 때에 데이터를 전송

Request : request method + URL + header + body




Response: Status code + header + body

# HTTP protocol + Rest API : 규약

- HTTP protocol 준수 + Rest이라는 어떤 특징을 가진 API (데이터 규칙)
  -> 서버 - 클라 / 서버 - 서버 간 소통을 함

- 요청의 종류 : GET, POST, PUT, DELETE, PATCH
- 데이터 전달의 방식
  - Header : 데이터를 보내는 사람, 인증 정보, 시간, IP, 나라 등등 부가적인 정보들
  - Body (주로 JSON) : 요청과 관련된 정보(parameter) / 요청 시 서버가 내려주는 응답 (response) -> Rest API면 Header에 Body를 판단할 수 있는 정보가 보통 포함됨

# iOS의 네트워킹 모듈

- URLSession

  : 퍼스트파티의 특성 상 기본 기능만을 구현 (물론 없는 건 없음)

  -> 에러 처리, 재시도, 인증을 위한 토큰 얹기 등의 부가작업이 귀찮음

  : 이거 그냥 써도 무방

- 하지만 불편하니까…

# 오픈소스 라이브러리의 활용

- 모든 코드를 혼자 다 짜는 것은 매우 비효율적!
- 누군가 짜놓은 코드가 내가 필요한 부분이라면 가져다 쓰는 것이 이득!

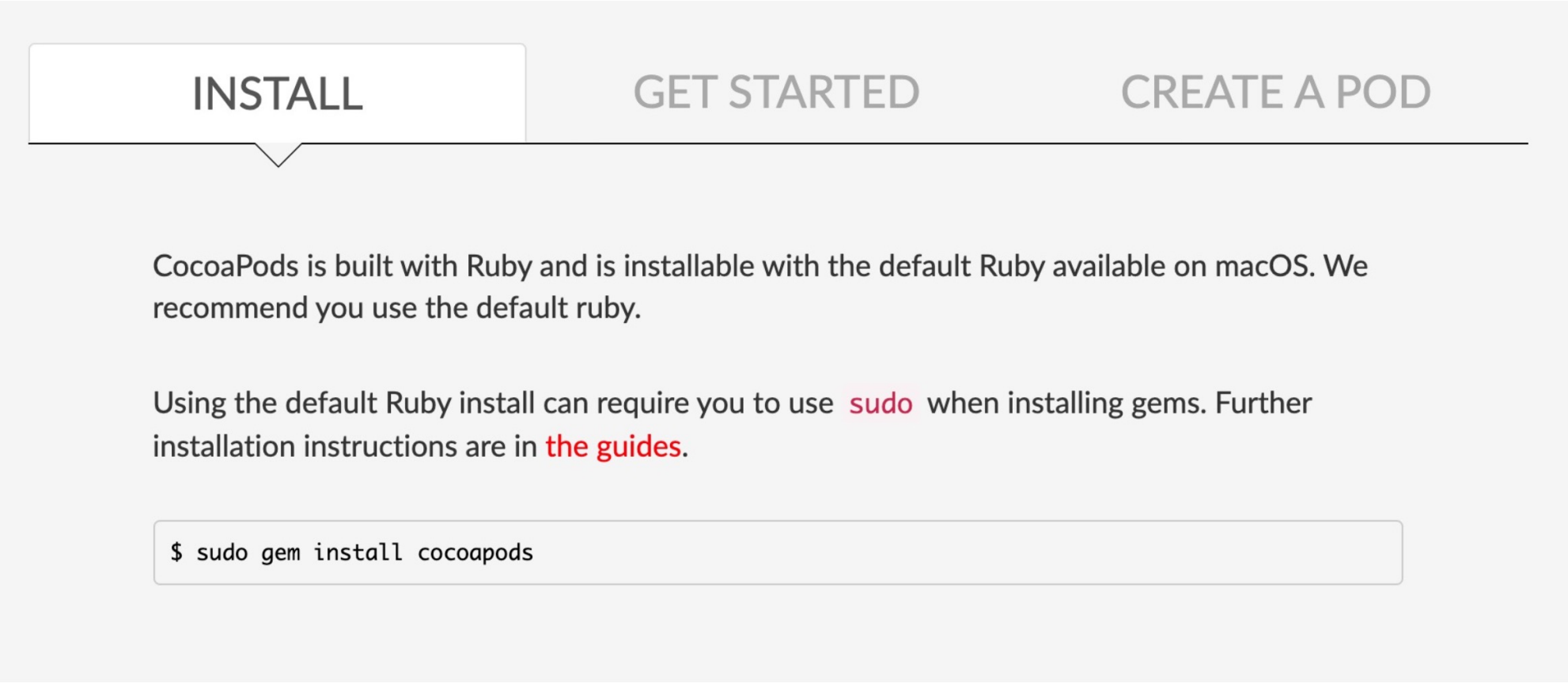
# 3rd party 관리 툴

- Cocoapods
- Swift Package Manager : Apple이 지원하는 공식 툴
- Carthage

- 우리는 일단 가장 만만한 Cocoapods를 쓸겁니다

  : SPM이 빌드 속도에도 긍정적이고 좋긴 한데 조금 어려워서..

# CocoaPods


INSTALL
GET STARTED
CREATE A POD
CocoaPods is built with Ruby and is installable with the default Ruby available on macOS. We recommend you use the default ruby.
Using the default Ruby install can require you to use sudo when installing gems. Further installation instructions are in the guides.
$ sudo gem install cocoapods

# CocoaPods

INSTALL | GET STARTED | CREATE A POD

Search for pods (above). Then list the dependencies in a text file named `Podfile` in your Xcode project directory:

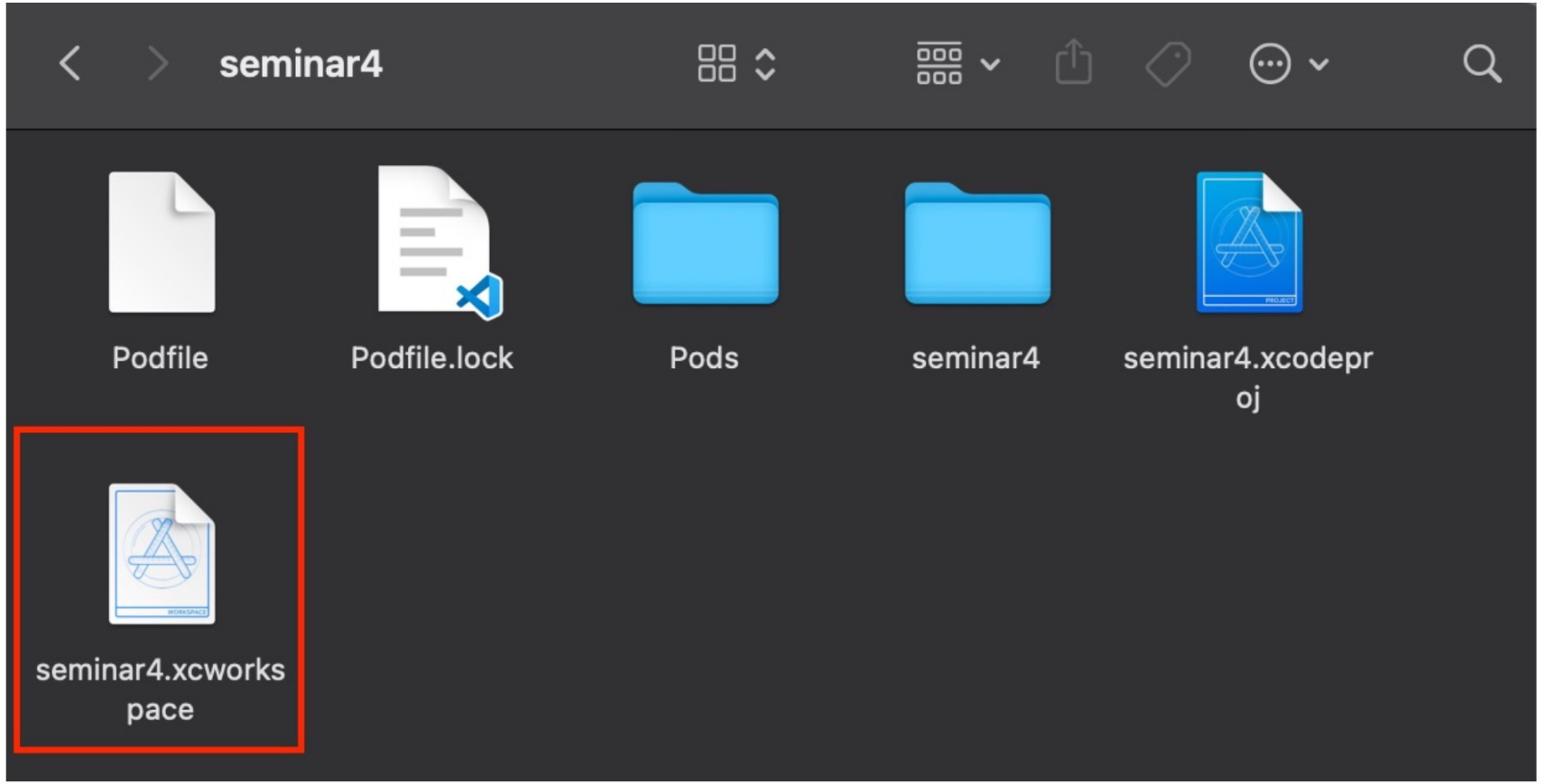
```
platform :ios, '8.0'
use_frameworks!

target 'MyApp' do
  pod 'AFNetworking', '~> 2.6'
  pod 'ORStackView', '~> 3.0'
  pod 'SwiftyJSON', '~> 2.3'
end
```

Tip: CocoaPods provides a `pod init` command to create a Podfile with smart defaults. You should use it.

Now you can install the dependencies in your project:

```
$ pod install
```

# 프로젝트의 변화?!


seminar4
Podfile
Podfile.lock
Pods
seminar4
seminar4.xcodeproj
seminar4.xcworkspace

# Alamofire의 사용

- 제일 확실한 건 공식 Document : https://github.com/Alamofire/Alamofire

```
struct Login: Encodable {
    let email: String
    let password: String
}

let login = Login(email: "test@test.test", password: "testPassword")

AF.request("https://httpbin.org/post",
           method: .post,
           parameters: login,
           encoder: JSONParameterEncoder.default).response { response in
    debugPrint(response)
}
```

# 네이버 오픈API

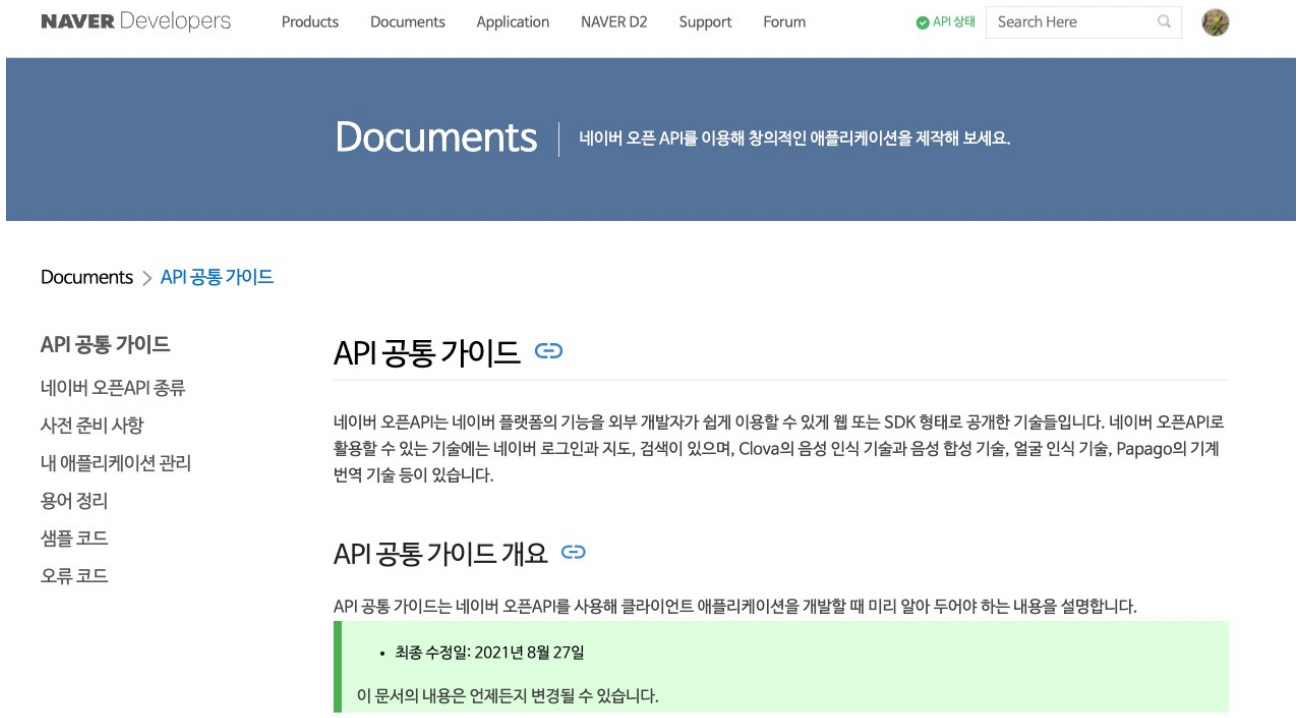
- 개발자들 쓰라고 공공기관 / 대기업들이 데이터를 제공해줍니다!

  : 잘 써먹으면 좋겠죠?

- 우리가 프로젝트에서 이런 오픈 API를 쓰게 되더라도 데이터 정합성을 위해 보통은 서버가 한 번 데이터를 정제해주지만…
- 지금은 서버가 없으니 그냥 직접 네이버에 데이터를 요청해보도록 합시다.

# 네이버 오픈API

- https://developers.naver.com/docs/common/openapiguide/



# 네이버 오픈API

- 앱 프로젝트를 먼저 생성 -> Bundle Identifier를 등록
- 획득한 secret 정보를 이용하여 API 요청하여 데이터 획득
- 획득한 데이터는 JSON 형식이기 때문에 Codable을 이용해 우리가 사용할 Model로의 변환이 필수적

# 코드 보면서 이야기 해봅시다

# 다음 시간 예고 : RxSwift + MVVM

```
let data = Observable<[String]>.just(["first element", "second element", "third element"])

data.bind(to: tableView.rx.items(cellIdentifier: "Cell")) { index, model, cell in
  cell.textLabel?.text = model
}
.disposed(by: disposeBag)
```

# Assignment 2 - 네이버 뉴스 헤드라인앱

- 검색 바에 키워드를 입력하고, 검색 버튼을 누르면 뉴스 헤드라인을 볼 수 있는 앱
- 뉴스는 검색 시 20개가 노출되며, 기사 제목, 작성 날짜가 표기되어야 함
- 기사 제목은 말줄임표 (…)으로 잘려서는 안 되며 특정 범위를 넘어가면 자동으로 개행이 되어 모든 내용을 표현해야 함
- 작성 날짜는 yyyy년 MM월 dd일의 형식으로 표현되어야 함
- 필수 : ViewController가 절대 뉴스 데이터를 가지고 있어서는 안 됨 -> ViewModel을 만들어서 서버 데이터 요청 + 저장 + ViewController에 전달하는 형태로 구현해야 함 (MVVM) -> 만족하지 않으면 돌아가도 탈락
  : 참고 -> 과제1 예제 코드

- 추가 기능 : Pagination -> 20개의 뉴스를 받아오고 나서 뉴스가 더 있을테니 스크롤을 끝까지 내리면 자동으로 다음 20개 또 다음 20개 화면에 보여줄 수 있도록 하는 기능

# Assignment 2 - 네이버 뉴스 헤드라인앱

- 개발에 필요한 지식

1. UITableView + UITableViewCell + Delegate, DataSource
2. Network (Header와 Body를 이용한 Rest API 요청 + Response 핸들링)
   : URLSession을 써도, Alamofire를 써도 무방. 성공적인 데이터 요청만 되면 됨
3. 오픈 API 활용법
4. MVVM
5. UITextField or UISearchBar

<추가 기능 개발 시 참고>

- iOS UITableView Pagination 검색해보시면 많이 나옵니다~

# 혼자 공부해보면 좋을 내용

- Cocoapods : https://cocoapods.org/
- Alamofire : https://github.com/Alamofire/Alamofire
- 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