Open API 기반 iOS앱 개발(2)

네트워킹

1. URL 만들기
   - URL(string: String)
2. URLSession 만들기
   - URLSession(configuration: URLSessionConfiguration)
3. URLSession 인스턴스에게 task주기
   - URLSession인스턴스.dataTask(with:completionHandler:)
4. task시작하기
   - URLSessionTask인스턴스.resume()

movieURL지정과 데이터 가져올 메서드 지정하고 호출

- getData()

```
8  class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
       @IBOutlet weak var table: UITableView!
10     let movieURL =
           "https://kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/searchDailyBoxOfficeList
           .json?key=408                                    541d7&targetDt=20210515"
11
12     override func viewDidLoad() {
13         super.viewDidLoad()
14         // Do any additional setup afte
15         table.delegate = self
16         table.dataSource = self
17         getData()
18     }
19     func getData(){
20
21     }
```

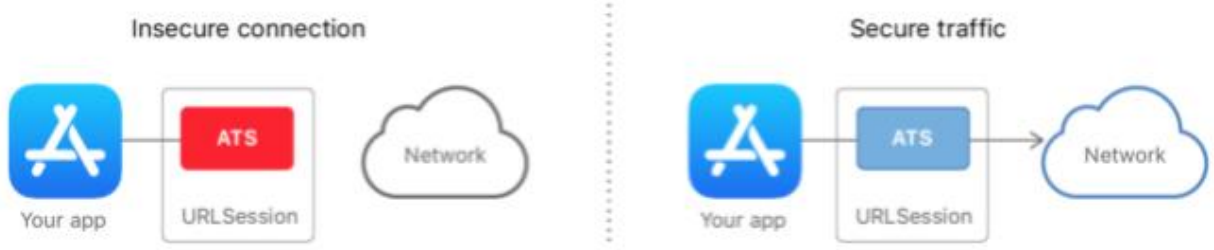
주의:http를 https로 변경

```
[redacted] 13:37:48.449519+0900 MovieHsh[29766:4021157] App Transport Security has blocked a cleartext HTTP (http://)
resource load since it is insecure. Temporary exceptions can be configured via your app's Info.plist file.
```

https프로코콜을 지원하지 않는 Open API를 사용할 경우에는 오류가 발생하여 다음 설정을 해야 함

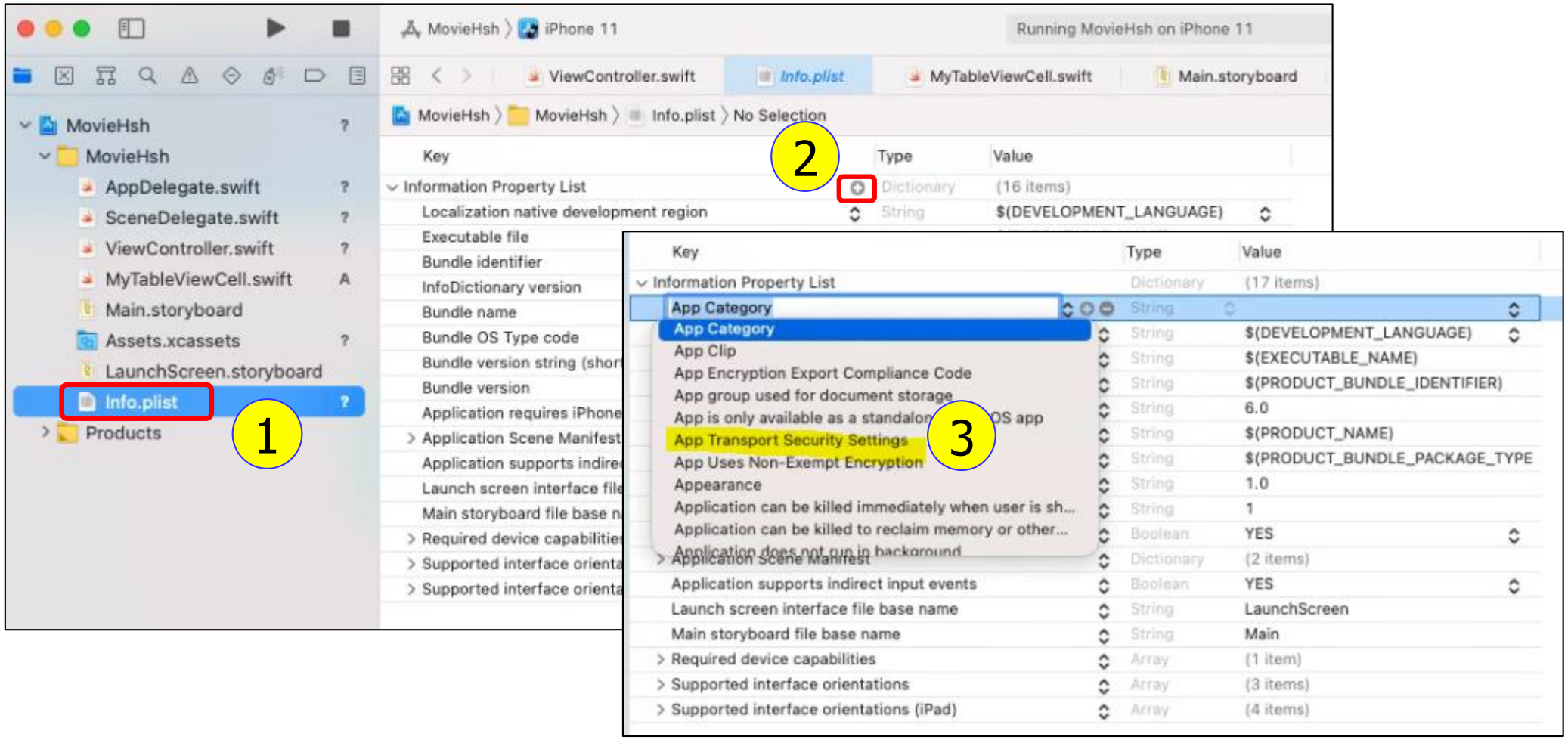
App Transport Security(ATS)

- https://developer.apple.com/documentation/security/preventing_insecure_network_connections
- iOS 9부터 외부 네트워크와 관련된 보안 규칙이 신설
- 네트워크 객체를 사용해서 SSL 보안 프로토콜을 사용하지 않는 네트워크에 접속하려면 info.plist 파일에 설정을 추가해주어야 함
- SSL 보안 프로토콜이 적용된 네트워크는 접속시 https:// 를 사용
  - 적용안된 네트워크는 http:// 을 사용함
- 서버 도메인이 https://로 시작하면 ATS 보안을 적용하지 않아도 됨
- http:// 로 시작하면 반드시 ATS 보안을 적용하여야 함



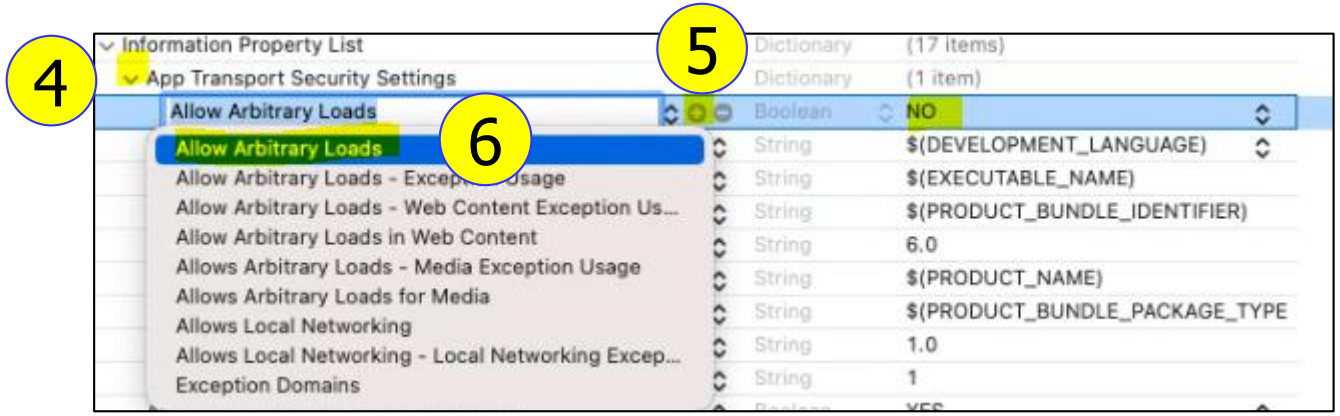
ATS를 위해 info.plist수정

- [+]눌러서 드롭 다운 목록에서 App Transport Security Settings 선택



Allow Arbitrary Loads : YES

- [Enter]키를 한번 더 누르고 [App Transport Security Settings]왼쪽 화살표가 아래로 향하도록 함
- 오른쪽 [+]버튼을 눌러 [Allow Arbitrary Loads] 선택하고 [엔터]키 누름
- [Value]를 [YES]로 변경



tableView(_:cellForRowAt:)은 얼마나 자주 호출될까요?

```
28      func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
29          let cell = tableView.dequeueReusableCell(withIdentifier: "myCell", for:
                MyTableViewCell
30          cell.movieName.text = indexPath.description //name[indexPath.row]
31          print(indexPath.description, separator: " ", terminator: " ")
32          return cell
33      }
34      func numberOfSections(in tableView: UITableView) -> Int {
35          return 5
36      }
37      func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPat
38          print(indexPath.description)
39      }
40  }
41
42
```

```
[0, 0] [0, 1] [0, 2] [0, 3] [0, 4] [1, 0] [1, 1] [1, 2] [1, 3] [1, 4] [2, 0] [2, 1] [
    2] [3, 3] [3, 4] [4, 0] [4, 1] [4, 2] [4, 3] [4, 4] [0, 4] [0, 3] [0, 2] [0, 1] [(
    3] [4, 4] [0, 4] [0, 3] [0, 2] [0, 1] [0, 0] [4, 0] [4, 1]
```

[1, 2]
[1, 3]
[1, 4]
[2, 0]
[2, 1]
[2, 2]
[2, 3]
[2, 4]
[3, 0]
[3, 1]
[3, 2]
[3, 3]
[3, 4]
[4, 0]

1급 객체(first class object)
1급 시민(first class citizen)

1급 객체(first class object) 또는 1급 시민(first class citizen)

- https://en.wikipedia.org/wiki/First-class_citizen
- Swift의 함수는 1급 객체이다.
  1급 객체(first class object) 또는 1급 시민(first class citizen)
- 다음 조건을 충족하는 객체를 1급 객체라고 한다.

1) 변수에 저장할 수 있다.
2) 매개변수로 전달할 수 있다.
3) 리턴값으로 사용할 수 있다.

Concept	Description	Languages
first-class function	closures and anonymous functions	Dart, Scheme, ML, Haskell, F#, Kotlin, Scala, Swift, Perl, PHP, Python, Raku, JavaScript, Delphi, Rust
first-class control	continuations	Scheme, ML, F#
first-class type	dependent types	Coq, Idris, Agda
first-class data type		Generic Haskell, C++11
first-class polymorphism	impredicative polymorphism	
first-class message	dynamic messages (method calls)	Smalltalk,[8] Objective-C[8]
first-class class	metaclass	Smalltalk, Objective-C, Ruby, Python, Delphi
first-class proofs	proof object[9]	Coq, Agda

함수 : 일급 객체 실습

```
func up(num: Int) -> Int {
   return num + 1
}
func down(num: Int) -> Int {
   return num - 1
}
let toUp = up
print(up(num:10))
print(toUp(10))
let toDown = down

func upDown(Fun: (Int) -> Int, value: Int) {
   let result = Fun(value)
   print("결과 = \(result)")
}
upDown(Fun:toUp, value: 10)   //toUp(10)
upDown(Fun:toDown, value: 10) //toDown(10)
```

(1) (1) (2) (2) (2)

```
func decideFun(x: Bool) -> (Int) -> Int {
                //매개변수형  리턴형이 함수형
  if x {
    return toUp
  } else {
    return toDown
  }
}
let r = decideFun(x:true) // let r = toUp
print(type(of:r))  //(Int) -> Int
print(r(10))  // toUp(10)
```

(3) (3) (3) (3)

Swift의 함수는 1급 객체이다.
1급 객체(first class object) 또는 1급 시민(first class citizen)
다음 조건을 충족하는 객체를 1급 객체라고 한다.
1) 변수에 저장할 수 있다.
2) 매개변수로 전달할 수 있다.
3) 리턴값으로 사용할 수 있다.

클로저(Closure)

클로저 표현식

- https://en.wikipedia.org/wiki/Closure_(computer_programming)
- 익명함수
- C, C++, Objective-C의 block
- Java의 Lambda function
- C#의 Delegates
- 클로저 표현식은 독립적인 코드 블록

```
func add(x: Int, y: Int) -> Int {
  return(x+y)
}
print(add(x:10, y:20))
```

```
let add1 = { (x: Int, y: Int) -> Int in
  return(x+y)
}
print(add1(x:10, y:20)) //주의 error: extraneous(관련 없는) argument labels 'x:y:' in call
print(add1(10, 20))     //OK
print(type(of:add1))    //과제
```

클로저 표현식

- 클로저 표현식은 매개변수를 받거나, 값을 반환하도록 만들 수도 있음

```
{(<매개변수 이름>: <매개변수 타입>, … ) -> <반환 타입> in
   // 클로저 표현식 코드
}
```

- 두 개의 정수 매개변수를 받아서 곱한 값을 반환

```
let multiply = {(val1: Int, val2: Int) -> Int  in
                        //매개변수          리턴형
        return val1 * val2
}// 여기서 multiply의 자료형은 (Int, Int) -> Int
let result = multiply(10, 20) //상수를 함수처럼 호출,200
```

```
func mul(val1: Int, val2: Int) -> Int
{
  return val1 * val2
}
let result = mul(val1:10, val2:20)
print(result)
```

후행 클로저(trailing closure)

- 클로저가 함수의 마지막 argument라면 마지막 매개변수명(cl)을 생략한 후 함수 소괄호 외부에 클로저를 작성
- https://docs.swift.org/swift-book/documentation/the-swift-programming-language/closures/

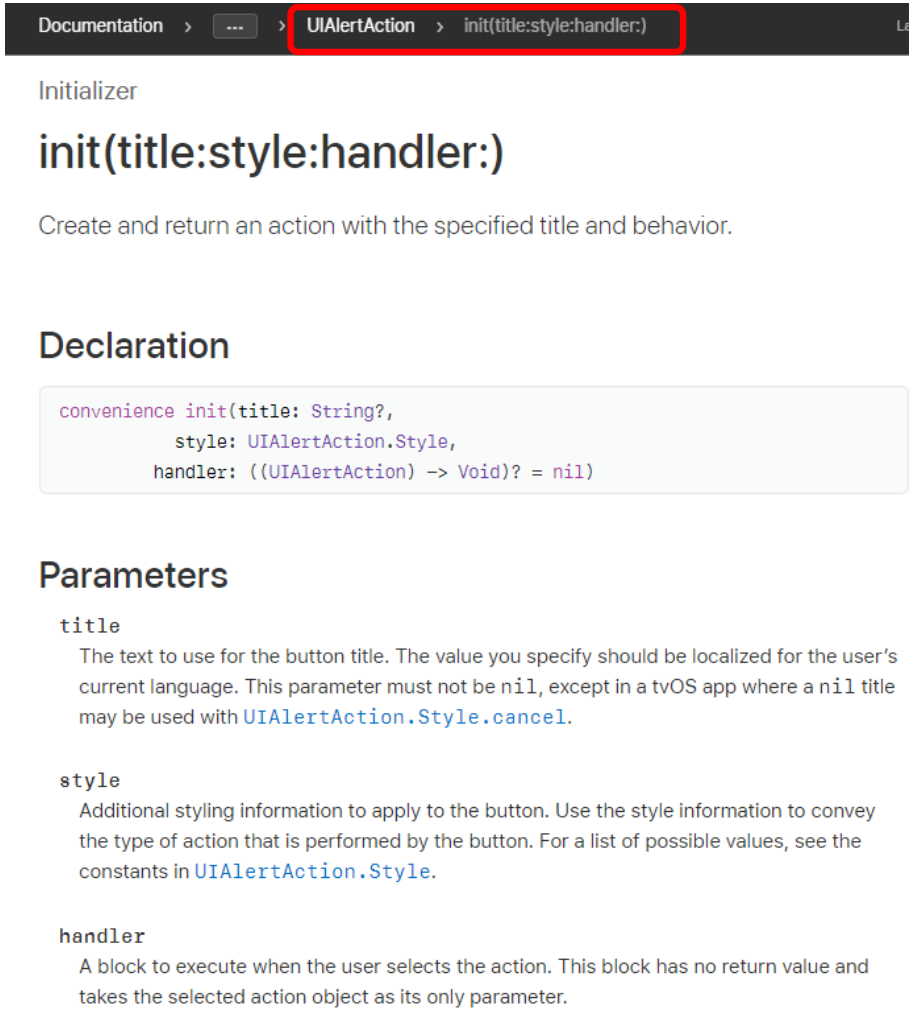
```
func someFun(cl: () -> Void) {

}
// trailing closure를 사용 안하면
someFun(cl: {
  //closure's body
})
// trailing closure 사용
someFun() {  //"cl:"을 생략하고 함수 소괄호 다음에 클로저 작성
  //trailing closure's body goes here
}
```

후행 클로저(trailing closure)

- 클로저가 함수의 마지막 argument라면 마지막 매개변수 이름(handler:)을 생략한 후 함수 소괄호 외부에 클로저를 구현
- https://docs.swift.org/swift-book/LanguageGuide/Closures.html

```
let onAction = UIAlertAction(title: "On", style:
 UIAlertAction.Style.default) {
   ACTION in self.lampImg.image = self.imgOn
   self.isLampOn=true
}
let removeAction = UIAlertAction(title: "제거", style:
  UIAlertAction.Style.destructive, handler: {
    ACTION in self.lampImg.image = self.imgRemove
    self.isLampOn=false
})
```

https://developer.apple.com/documentation/uikit/uialertaction/1620097-init



클로저의 축약 표현들

```
let multiply = {(val1: Int, val2: Int) -> Int  in
        return val1 * val2
}
var result = multiply(10, 20)
print(result)
```

```
func mul(val1: Int, val2: Int) -> Int
{
  return val1 * val2
}
let result = mul(val1:10, val2:20)
```

```
let add = {(val1: Int, val2: Int) -> Int  in
        return val1 + val2
}
result = add(10, 20)
print(result)
```

```
func math(x: Int, y: Int, cal: (Int, Int) -> Int) -> Int {
    return cal(x, y)
}
result = math(x: 10, y: 20, cal: add)
print(result)
result = math(x: 10, y: 20, cal: multiply)
print(result)
```

```
result = math(x: 10, y: 20, cal: {(val1: Int, val2: Int) -> Int  in
        return val1 + val2
}) //클로저 소스를 매개변수에 직접 작성
print(result)
```

```
result = math(x: 10, y: 20) {(val1: Int, val2: Int) -> Int  in
        return val1 + val2
}//trailing closure
print(result)
```

클로저가 함수의 마지막 argument라면 마지막 매개변수 이름(cal:)을 생략한 후 함수 소괄호 외부에 클로저를 구현

```
result = math(x: 10, y: 20, cal: {(val1: Int, val2: Int)   in
        return val1 + val2
}) //리턴형 생략
print(result)
```

```
result = math(x: 10, y: 20) {(val1: Int, val2: Int)  in
        return val1 + val2
} //trailing closure, 리턴형 생략
print(result)
```

```
result = math(x: 10, y: 20, cal: {
        return $0 + $1
}) //매개변수 생략하고 단축인자(shorthand argument name)사용
print(result)
```

```
result = math(x: 10, y: 20) {
        return $0 + $1
} //trailing closure, 매개변수 생략하고 단축인자사용
print(result)
```

```
result = math(x: 10, y: 20, cal: {
         $0 + $1
}) //클로저에 리턴값이 있으면 마지막 줄을 리턴하므로 return생략
print(result)
```

```
result = math(x: 10, y: 20) { $0 + $1 } //return 생략
print(result)
```

디폴트 매개변수(아규먼트) 정의하기

- argument로 전달하는 값이 없는 경우, 디폴트 매개변수 값을 사용
- 함수를 선언할 때 매개변수에 디폴트 값을 할당
- 이름이 인자로 전달되지 않을 경우에 디폴트로 "길동"이라는 문자열이 사용되도록 함

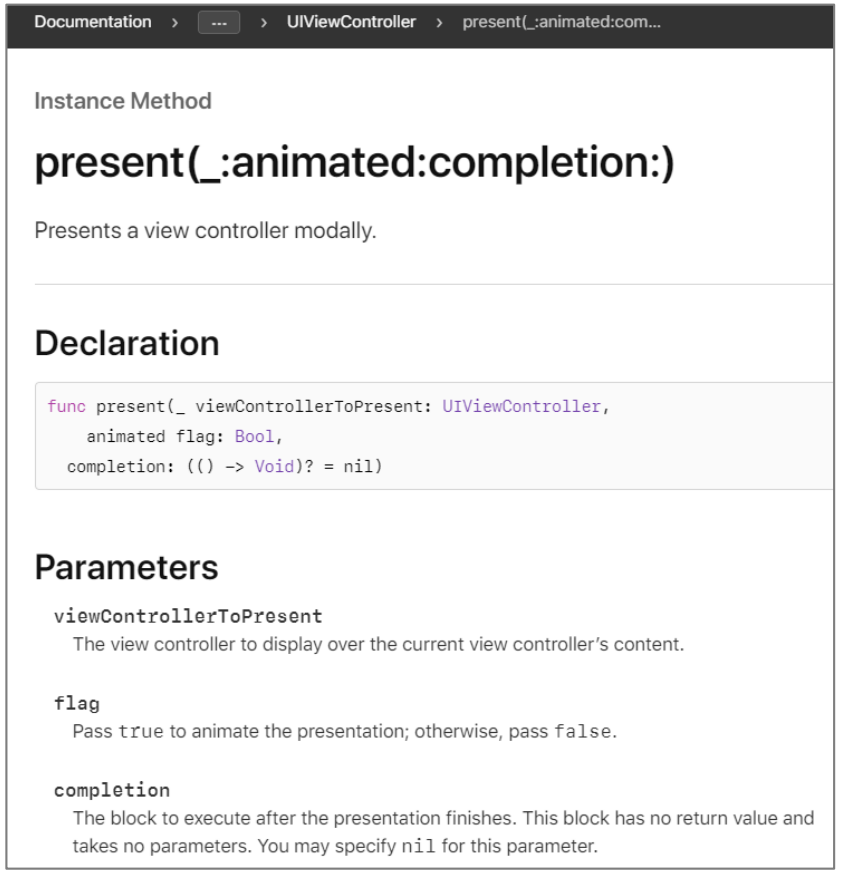
```
func sayHello(count: Int, name: String = "길동") -> String {
  return ("\(name) 번호는 \(count)")
}
var message = sayHello(count:10, name: "철수")
print(message) //철수 번호는 10
message = sayHello(count:100)   //name에 값을 전달하지 않음
print(message) //길동 번호는 100
```

present(_:animated:completion:)

- https://developer.apple.com/documentation/uikit/uiviewcontroller/1621380-present

```
func present(
    _ viewControllerToPresent: UIViewController,
    animated flag: Bool,
    completion: ( () -> Void)? = nil
)
```

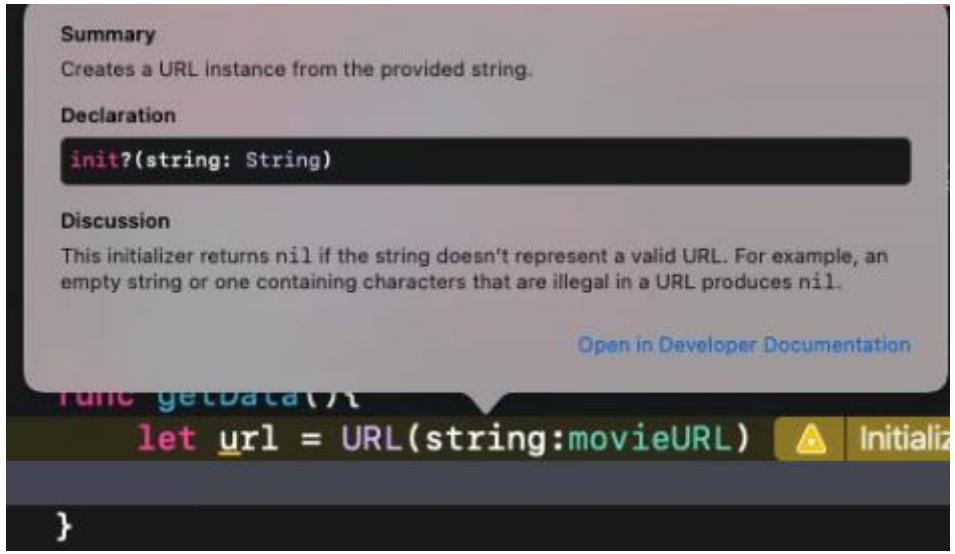
- 모달 방식으로 새로운 뷰콘트롤러를 보여 줌
  - 모달: 이 뷰콘트롤러를 없애야 원래(아래)뷰콘트롤러를 볼 수 있음
  1. 현재 view controller 위에 표시할 view controller
  2. true는 새로운 뷰를 애니메이션하면서 보여주고, false는 안함
  3. 새로운 뷰 콘트롤러가 보여진 후 실행할 블록. 여기에 리턴값이 없으며 매개 변수도 없는 클로저 작성. 특별히 하고 싶은 일이 없다면 nil을 지정하거나 디폴트 인자가 nil이므로 생략 가능
- present(alert, animated: true, completion: nil)
- present(alert, animated: true)



네트워킹 1단계 : URL 만들기

1. URL 만들기
2. URLSession 만들기
3. URLSession 인스턴스에게 task주기
4. task시작하기(task.resume())

- init?(string: String)
- failable initializer이므로 url을 옵셔널 바인딩
- https://developer.apple.com/documentation/foundation/url




```
func getData(){
    if let url = URL(string:movieURL){

    }

}
```

네트워킹 2단계 : URLSession 만들기

1. URL 만들기
2. URLSession 만들기
3. URLSession 인스턴스에게 task주기
4. task시작하기(task.resume())

- https://developer.apple.com/documentation/foundation/urlsession

```
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
```

Documentation › Foundation › URL Loading System › URLSession

Class

URLSession

An object that coordinates a group of related, network data-transfer tasks.

Declaration

```
class URLSession : NSObject
```

Overview

The URLSession class and related classes provide an API for downloading data from and uploading data to endpoints indicated by URLs. Your app can also use this API to perform background downloads when your app isn't running or, in iOS, while your app is suspended. You can use the related URLSessionDelegate and URLSessionTaskDelegate to support authentication and receive events like redirection and task completion.

열거형(enum)

Documentation › ... › URLSession › init(configuration:)

Initializer

init(configuration:)

Creates a session with the specified session configuration.

네트워킹 3단계 : URLSession 인스턴스에게 task주기

1. URL 만들기
2. URLSession 만들기
3. URLSession 인스턴스에게 task주기
4. task시작하기(task.resume())

- dataTask(with:completionHandler:)
- https://developer.apple.com/documentation/foundation/urlsession/1410330-datatask
- 지정된 URL의 내용을 검색하는 작업을 만든(create) 다음, **완료시** handler(클로저)를 호출
- https://docs.swift.org/swift-book/LanguageGuide/Closures.html
- 클로저 앞에 @escaping 가 있으면 함수의 작업이 완료된 후에 클로저가 호출됨
  - func dataTask(
    with url: URL,
    completionHandler: @escaping (Data?, URLResponse?, Error?) -> Void
) -> URLSessionDataTask
  - 매개 변수
    - url
    - completionHandler : 로드 요청이 완료되면 호출할 클로저
      - data : 서버에서 반환된 데이터
      - response : HTTP 헤더 및 상태 코드와 같은 응답 메타 데이터를 제공하는 객체
      - error: 요청이 실패한 이유
  - 리턴값
    - 새 세션 데이터 task
  - 작업(task)을 만든 후에는 resume()메서드를 호출하여 시작해야함

dataTask(with:completionHandler:)

- https://developer.apple.com/documentation/foundation/urlsession/1410330-datatask
- 지정된 URL의 내용을 검색하는 작업을 만든(create) 다음, 완료시 handler(클로저)를 호출

```
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
        session.dat                                   ⊗ Value of type 'URLSession' has no member 'd
    }
}
func tableVi
    return na
```

```
M dataTask(with request: URLRequest)
M dataTask(with url: URL)
M dataTaskPublisher(for url: URL)
M dataTaskPublisher(for request: URLRequest)
M dataTask(with request: URLRequest, completionHandler: (Data?, URLResponse?, Error?
M dataTask(with url: URL, completionHandler: (Data?, URLResponse?, Error?) -> Void)
```

```
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
        session.dataTask(with: url, completionHandler: (Data?, URLResponse?, Error?) -> Void)

    }
}
```

여기서 enter

```
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
        session.dataTask(with: url) { (Data?, URLResponse?, Error?) in
            code
        }

    }
}
```

후행 클로저 스타일로 자동으로 바뀜

소스 수정하고 실행했는데 동작 안함

- completionHandler부분을 후행 클로저로 작성

```
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
        session.dataTask(with: url) { (data, response, error) in
            if error != nil {
                print(error!)
                return
            }
            if let JSONdata = data{
                print(JSONdata)
            }
        }

    }
```

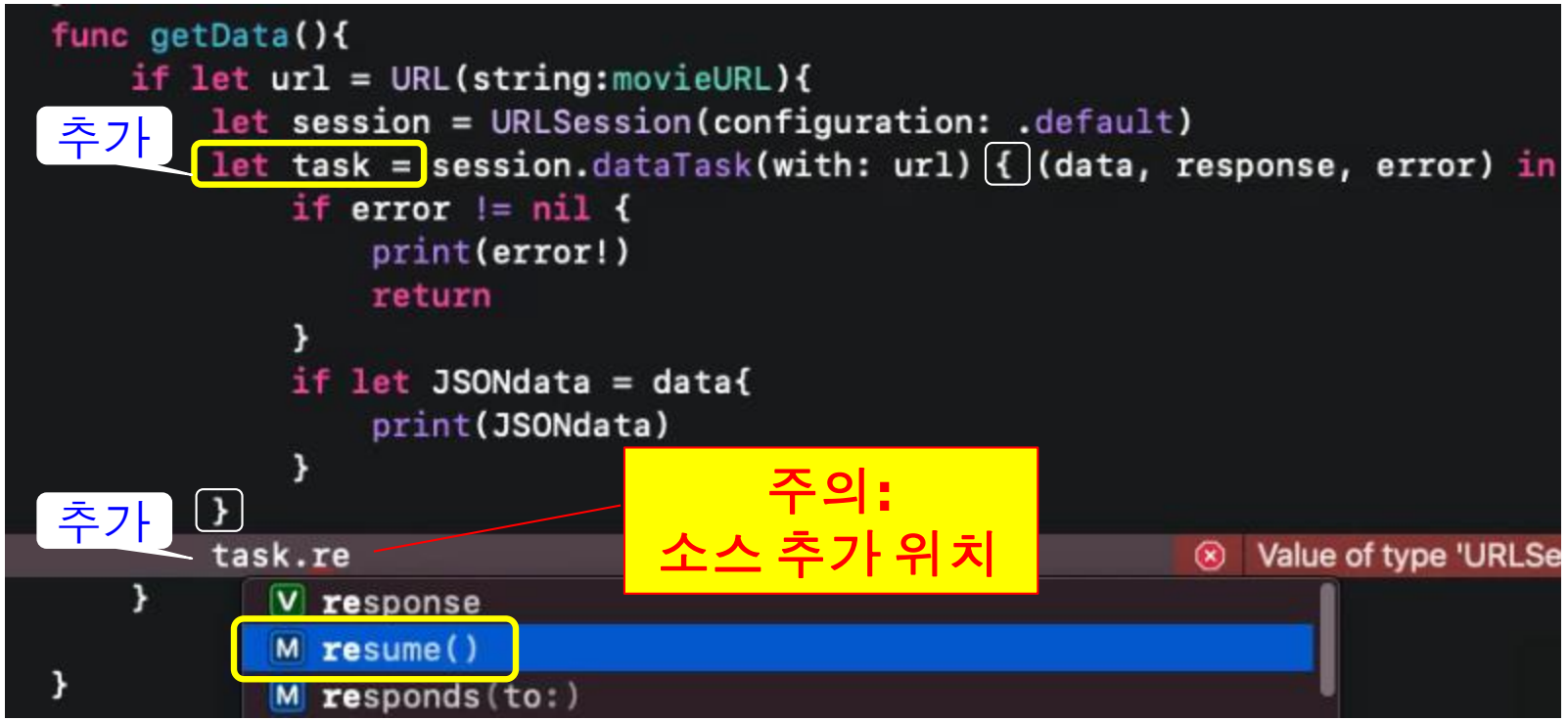
옵셔널 형이라서
옵셔널 바인딩

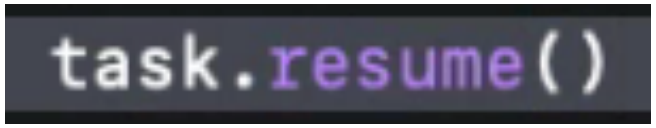
받아온 바이트 수 출력

네트워킹 4단계 : task를 resume()

- https://developer.apple.com/documentation/foundation/urlsessiontask/1411121-resume
- 작업이 일시 중단된 경우 다시 시작하는 메서드
- 새로 초기화된 작업은 일시 중단된 상태에서 시작되므로, 이 메서드를 호출하여 작업을 시작해야함

1. URL 만들기
2. URLSession 만들기
3. URLSession 인스턴스에게 task주기
4. task시작하기(task.resume())





과제 : response에는 어떤 정보가?

- https://developer.apple.com/documentation/foundation/urlsession/1410330-datatask

Declaration

```
func dataTask(with url: URL,
completionHandler: @escaping (Data?, URLResponse?, Error?) -> Void) -> URLSessionDataTask
```

Parameters

`url`
The URL to be retrieved.

`completionHandler`
The completion handler to call when the load request is complete. This handler is executed on the delegate queue.

If you pass `nil`, only the session delegate methods are called when the task completes, making this method equivalent to the `dataTask(with:)` method.

This completion handler takes the following parameters:

`data`
The data returned by the server.

`response`
An object that provides response metadata, such as HTTP headers and status code. If you are making an HTTP or HTTPS request, the returned object is actually an `HTTPURLResponse` object.

`error`
An error object that indicates why the request failed, or `nil` if the request was successful.

Return Value

The new session data task.

```
<NSHTTPURLResponse: 0x6000009b0960> { URL:
    https://kobis.or
    .kr/kobisopenapi/webservice/rest/boxoffice/searchDailyBoxOffic
    eList
    .json?key=4                                    '&targetDt=20220514
    } { Status Code: 200, Headers {
    Connection =     (
        "keep-alive"
    );
    "Content-Type" =     (
        "application/json;charset=utf-8"
    );
    Date =     (
        "Sun, 22 May 2022 15:03:28 GMT"
    );
    Server =     (
        "."
    );
    "Transfer-Encoding" =     (
        Identity
    );
} }
```

data를 String형식으로 찍어 보기

- 과연 잘 왔을까요?

```
let dataString = String(data: JSONdata, encoding: .utf8)
print(dataString!) //데이터를 스트링으로 찍어보기
```


```
{"boxOfficeResult":{"boxofficeType":"일별
    박스오피스","showRange":"20220514~20220514"
    ,"dailyBoxOfficeList":
    [
    {"rnum":"1","rank":"1","rankInten":"0","rankOldAndNew":"OLD"
    ,"movieCd":"20212855","movieNm":"닥터 스트레인지: 대혼돈의
    멀티버스","openDt":"2022-05-04","salesAmt":"4050572820"
    ,"salesShare":"72
    .3","salesInten":"2127707900","salesChange":"110
    .7","salesAcc":"49190240260","audiCnt":"369429"
    ,"audiInten":"193903","audiChange":"110
    .5","audiAcc":"4612535","scrnCnt":"2495","showCnt":"11403"},
    {"rnum":"2","rank":"2","rankInten":"0","rankOldAndNew":"NEW"
    ,"movieCd":"20204548","movieNm":"범죄도시
    2","openDt":"2022-05-18","salesAmt":"794804440"
    ,"salesShare":"14
    .2","salesInten":"794804440","salesChange":"100"
    ,"salesAcc":"904994440","audiCnt":"72986","audiInten":"72986"
    ,"audiChange":"100","audiAcc":"84187","scrnCnt":"459"
    ,"showCnt":"859"},
```

파싱을 쉽게 하기 위한 MovieData형 구조체 만들기

```swift
import UIKit
let name = ["aaa", "bbb","ccc","ddd","eee"]
struct MovieData : Codable {
    let boxOfficeResult : BoxOfficeResult
}
struct BoxOfficeResult : Codable {
    let dailyBoxOfficeList : [DailyBoxOfficeList]
}
struct DailyBoxOfficeList : Codable {
    let movieNm : String
    let audiCnt : String
}
class ViewController: UIViewController, UITableViewDelegate, UITabl
                                        table: UITableView!
```

주의:Codable 프로토콜 준수해야 함

1

2

3

```
▼ object {1}
▼ boxOfficeResult {3}
    boxofficeType : 일별 박스오피스
    showRange : 20220514~20220514
▼ dailyBoxOfficeList [10]
  ▼ 0 {18}
      rnum : 1
      rank : 1
      rankInten : 0
      rankOldAndNew : OLD
      movieCd : 20212855
      movieNm : 닥터 스트레인지:
```

Summary

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.


```
{
  "boxOfficeResult": {
    "boxofficeType": "일별 박스오피스",
    "showRange": "20220514~20220514",
    "dailyBoxOfficeList": [
      {
        "rnum": "1",
        "rank": "1",
        "rankInten": "0",
        "rankOldAndNew": "OLD",
        "movieCd": "20212855",
        "movieNm": "닥터 스트레인지:
        "openDt": "2022-05-04",
        "salesAmt": "4050678820",
        "salesShare": "72.3",
        "salesInten": "2127489900",
        "salesChange": "110.6",
        "salesAcc": "49191168260",
        "audiCnt": "369436",
        "audiInten": "193888",
        "audiChange": "110.4",
        "audiAcc": "4612609",
        "scrnCnt": "2495",
        "showCnt": "11403"
      },
      {
        "rnum": "2",
        "rank": "2",
        "rankInten": "0",
        "rankOldAndNew": "NEW",
        "movieCd": "20204548",
        "movieNm": "범죄도시 2",
        "openDt": "2022-05-18",
        "salesAmt": "794804440",
        "salesShare": "14.2",
        "salesInten": "794804440",
        "salesChange": "100",
```


응답 구조

응답 필드	값	
boxofficeType	문자열	박스오피스 종류를 출력합니다
showRange	문자열	박스오피스 조회 일자를 출력
rnum	문자열	순번을 출력합니다.
rank	문자열	해당일자의 박스오피스 순위를
rankInten	문자열	전일대비 순위의 증감분을 출력
rankOldAndNew	문자열	랭킹에 신규진입여부를 출력합 "OLD" : 기존 , "NEW" : 신규
movieCd	문자열	영화의 대표코드를 출력합니다
movieNm	문자열	영화명(국문)을 출력합니다.
openDt	문자열	영화의 개봉일을 출력합니다.
salesAmt	문자열	해당일의 매출액을 출력합니다
salesShare	문자열	해당일자 상영작의 매출총액 대
salesInten	문자열	전일 대비 매출액 증감분을 출
salesChange	문자열	전일 대비 매출액 증감 비율을
salesAcc	문자열	누적매출액을 출력합니다.
audiCnt	문자열	해당일의 관객수를 출력합니다
audiInten	문자열	전일 대비 관객수 증감분을 출
audiChange	문자열	전일 대비 관객수 증감 비율을
audiAcc	문자열	누적관객수를 출력합니다.
scrnCnt	문자열	해당일자에 상영한 스크린수를
showCnt	문자열	해당일자에 상영된 횟수를 출력

```
decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm
```

JSONDecoder : JSON객체에서 데이터 타입의 인스턴스를 디코딩

- https://developer.apple.com/documentation/foundation/jsondecoder
- func decode<T>(_ type: T.Type, from data: Data) throws -> T where T : Decodable

```
if let JSONdata = data{ //data가 옵셔널형임
    print(JSONdata,response!)
    let dataString = String(data: JSONdata, encoding: .utf8)
    print(dataString!) //데이터를 스트링으로 찍어보기
    let decoder = JSONDecoder()
    decoder.
        V dataDecodingStrategy
        V dateDecodingStrategy
        M decode(_:from:) throws
```

```
6  import UIKit
7  let name = ["aaa", "bbb","ccc","ddd","eee"]
8  struct MovieData : Codable {
9      let boxOfficeResult : BoxOfficeResult
10 }
11 struct BoxOfficeResult : Codable {
12     let dailyBoxOfficeList : [DailyBoxOfficeList]
13 }
14 struct DailyBoxOfficeList : Codable {
15     let movieNm : String
16     let audiCnt : String
17 }
```

```
decoder.decode(MovieData, from: JSONdata)  3  Call can throw, but it is not marked with 'try'...
```

```
decoder.decode(MovieData, from: JSONdata)
()
    Call can throw, but it is not marked with 'try' and the error is not handled
    Expected member name or constructor call after type name
    Add arguments after the type to construct a value of the type   Fix
    Use '.self' to reference the type object   Fix
```

JSON객체 JSONdata를 MovieData형으로 디코딩

메타 타입 : String.Type vs String.self

- func decode<T>(_ type: T.Type, from data: Data) throws -> T where T : Decodable

```
decoder.decode(MovieData.self, from: JSONdata)   2 ⚠ ⊗   Call can throw, but it is not marked wi...
```

- let x : String = "Hello"
  - String이 type이고 "Hello"가 x instance의 value

타입	타입 인스턴스 값
String	"Hello"
메타타입	메타타입 인스턴스 값
String.Type	String.self

```
let x : String = "Hello"
let y : String.Type = String.self
print(x,type(of:x)) // Hello String
print(y,type(of:y)) // String String.Type
print(String.self,type(of:String.self)) // String String.Type
print(Int.self,type(of:Int.self))       // Int Int.Type
```

```
class Man{
  var age : Int = 10
  var weight : Double = 35.5
  static var birthAge : Int = 1
  func display(){
   print("나이=\(age), 몸무게=\(weight)")
  }
  class func cM(){
         print("cM은 오버라이드가능한 클래스 메서드")
  }
  static func scM(){
         print("scM은 클래스 메서드(static)")
  }
}
class Student : Man {
    override class func cM(){
         print("cM은 오버라이드가능한 클래스 메서드")
  }
}
```

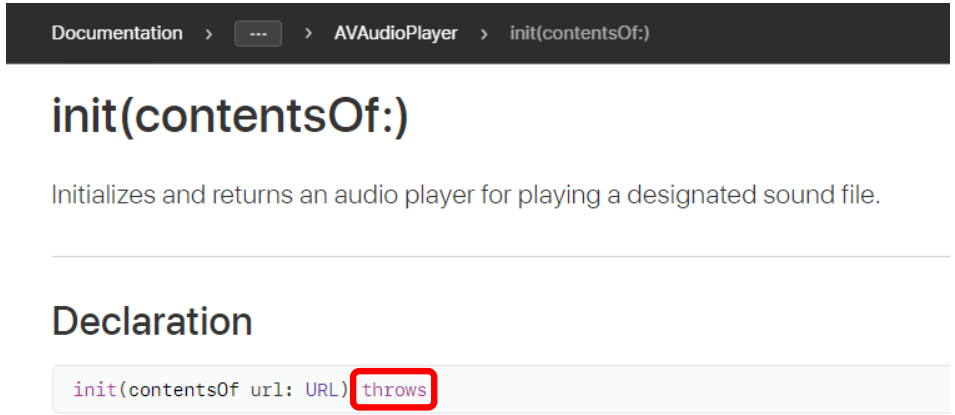
```
var kim : Man = Man()
print(Man.birthAge) //static property는 클래스가 호출
print(kim.age) //property는 인스턴스가 호출
kim.display() //인스턴스 메서드는 인스턴스가 호출
Man.cM()  //클래스 메서드는 클래스가 호출
Man.scM() //클래스 메서드는 클래스가 호출
var han : Student = Student()
print(type(of:han))
type(of: han).cM()
Student.cM()
var lee : Man = Student()  //lee는 컴파일시 Man형
print(type(of:lee))   //lee는 실행시 Student형
type(of: lee).cM()
```

Swift에서 오류 처리

- https://docs.swift.org/swift-book/documentation/the-swift-programming-language/errorhandling/
- 예외처리(exception handling)
- 런타임 시 오류를 발견하여 응답하고 복구하는 과정
- Swift에서는 optional을 사용하여 값의 유무를 전달함으로써 작업의 성공/실패 유무를 판단할 수 있지만 작업이 실패할 때 코드가 적절히 응답할 수 있도록 함으로써 오류의 원인을 이해하는 데 도움을 줄 수 있다.
- 디스크상의 파일을 읽어서 처리하는 작업에서 발생할 수 있는 오류
  - '존재하지 않는 파일', '읽기 권한 없음', '호환되는 형식이 아님' 등 다양
  - 오류의 원인에 따라 다양한 대응이 필요한 경우, 오류의 정보를 정확히 전달함으로써 오류를 복구하는데 도움을 줄 수 있음
- Swift 2.0 이후부터는 error handling을 도입

throwing function

- To indicate that a function, method, or initializer can throw an error, you write the throws keyword in the function's declaration after its parameters.
- A function marked with throws is called a throwing function.
- 매개변수 괄호 다음에 throws라는 키워드가 있는 함수는 그냥 사용할 수 없고 error handing을 해야 함
- func can() throws
  - 리턴값이 없는 throwing function
- func canThrowErrors() throws -> String
  - error handing을 해야하는 함수
- func cannotThrowErrors() -> String
  - error handing할 수 없는 함수

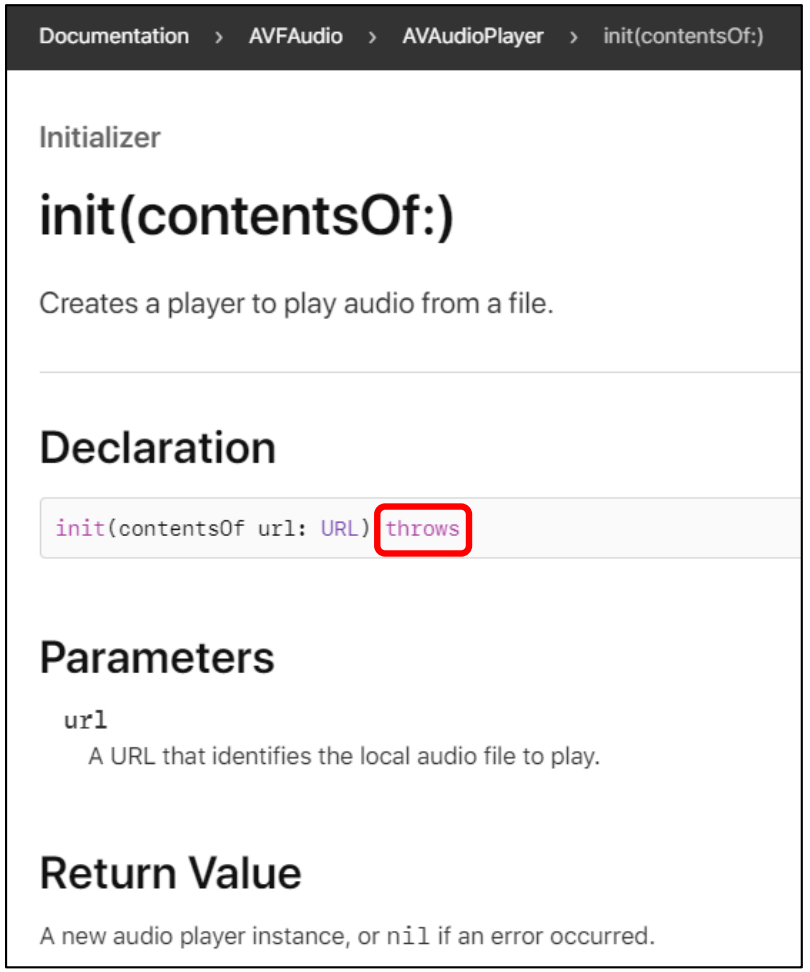


오류 발생 가능 함수의 호출 방식(do~try~catch)

```
do {
    audioPlayer = try AVAudioPlayer(contentsOf: audioFile)
} catch let error as NSError {
    print("Error-initPlay : \(error)")
}
```

- 이렇게 그냥 호출할 수는 없음
  - AVAudioPlayer(contentsOf: audioFile)
- do~try~catch로 error handing해야 함
  - 하지 않으면 Call can throw, but it is not marked with 'try' and the error is not handled" 오류가 발생

https://developer.apple.com/documentation/avfoundation/avaudioplayer



do~catch을 이용한 error handling

```
do {
    try 오류 발생 코드
    오류가 발생하지 않으면 실행할 코드
} catch 오류패턴1 {
    처리 코드
} catch 오류패턴2 where 조건 {
    처리 코드
} catch {
    처리 코드
}
```

에러 핸들링

- https://developer.apple.com/documentation/foundation/jsondecoder
- func decode<T>(_ type: T.Type, from data: Data) throws -> T where T : Decodable

```
if let JSONdata = data{ //data가 옵셔널형임
    print(JSONdata,response!)
    let dataString = String(data: JSONdata, encoding: .utf8)
    print(dataString!) //데이터를 스트링으로 찍어보기
    let decoder = JSONDecoder()
3   do{
        let decodedData = try 2 decoder.decode(MovieData.self, from: JSONdata) 1
        print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm)

    }catch{ 4
        print(error)
    }
```

do를 치면, Xcode 자동 완성 기능으로 Do-Catch문을 만들 수도 있지만, 여기서는 수동으로 작성함

소스정렬
[Editor]-[Structure]-[Re-Indent]

MovieData형 프로퍼티 만들어 decodedData 저장

- tableView(_:cellForRowAt:)에서 decodeData를 사용하기 위해

```
18 class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
      @IBOutlet weak var table: UITableView!
20    var movieData : MovieData?
21    let movieURL =
```

초기값이 없으므로 옵셔널형으로 지정

```
do{
  let decodedData = try decoder.decode(MovieData.self, from: JSONdata)
    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm)
    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].audiCnt)
    movieData = decodedData

}catch{
    print(error)
}
```

```
    movieData = decodedData

}catch{
    print(err
}
```

Reference to property 'movieData' in closure requires explicit use of 'self' to make capture semantics explicit

Reference 'self.' explicitly — Fix

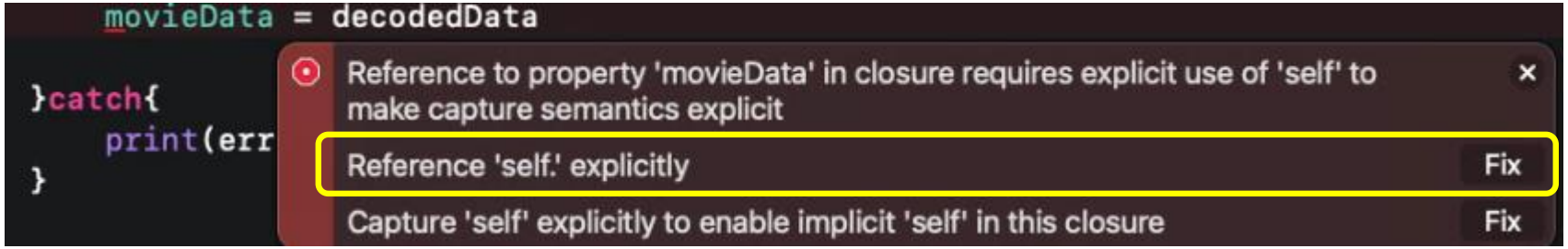
Capture 'self' explicitly to enable implicit 'self' in this closure — Fix

```
self.movieData = decodedData
```

클로저 내부에서 **property**접근시 **self.**써야 함

closure 내부에서 self. 사용하는 경우

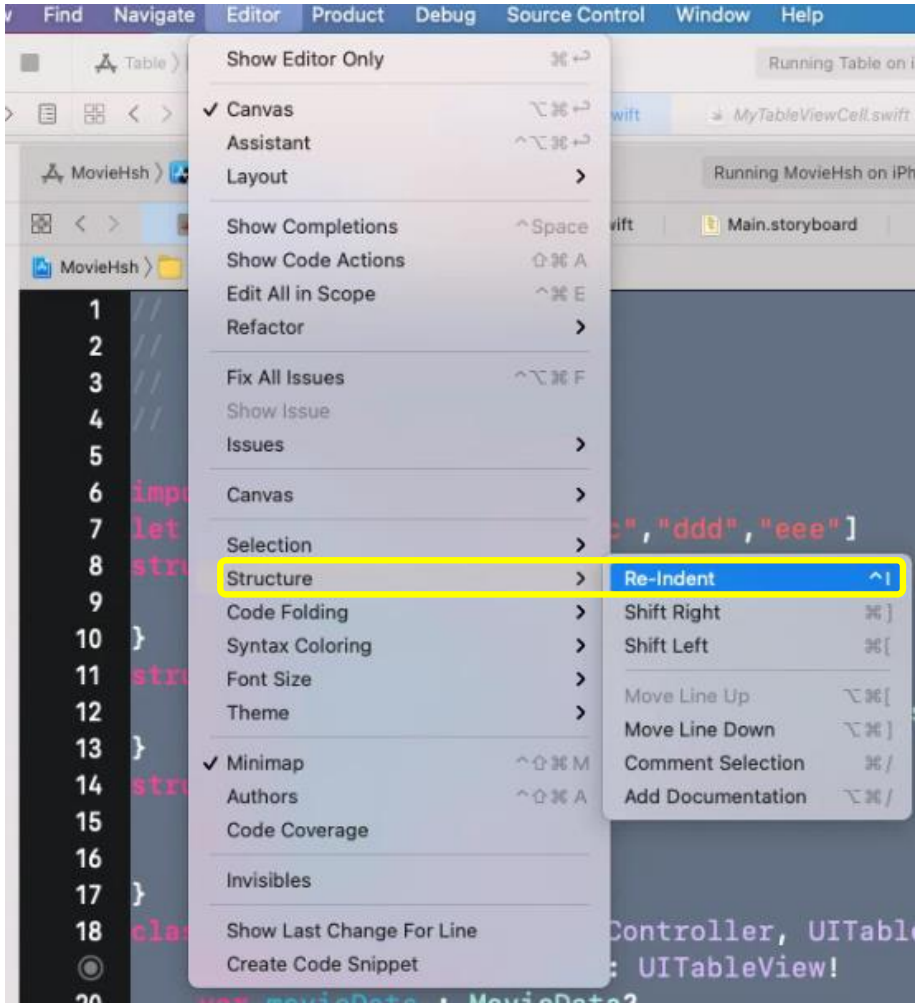
- https://docs.swift.org/swift-book/ReferenceManual/Expressions.html#ID544
- closure 안에서 객체의 변수 또는 함수에 접근할 때 self를 붙여서 사용




```
self.movieData = decodedData
```

소스 정렬

- 전체 소스 선택하고 [Editor]-[Structure]-[Re-Indent]



실행하면 결과가 좀 이상?

- tableView(_:cellForRowAt:) 수정

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "myCell", for: indexPath) as!
        MyTableViewCell
    //cell.movieName.text = name[indexPath.row]
    cell.movieName.text = movieData?.boxOfficeResult.dailyBoxOfficeList[indexPath.row].movieNm
    return cell
}
```

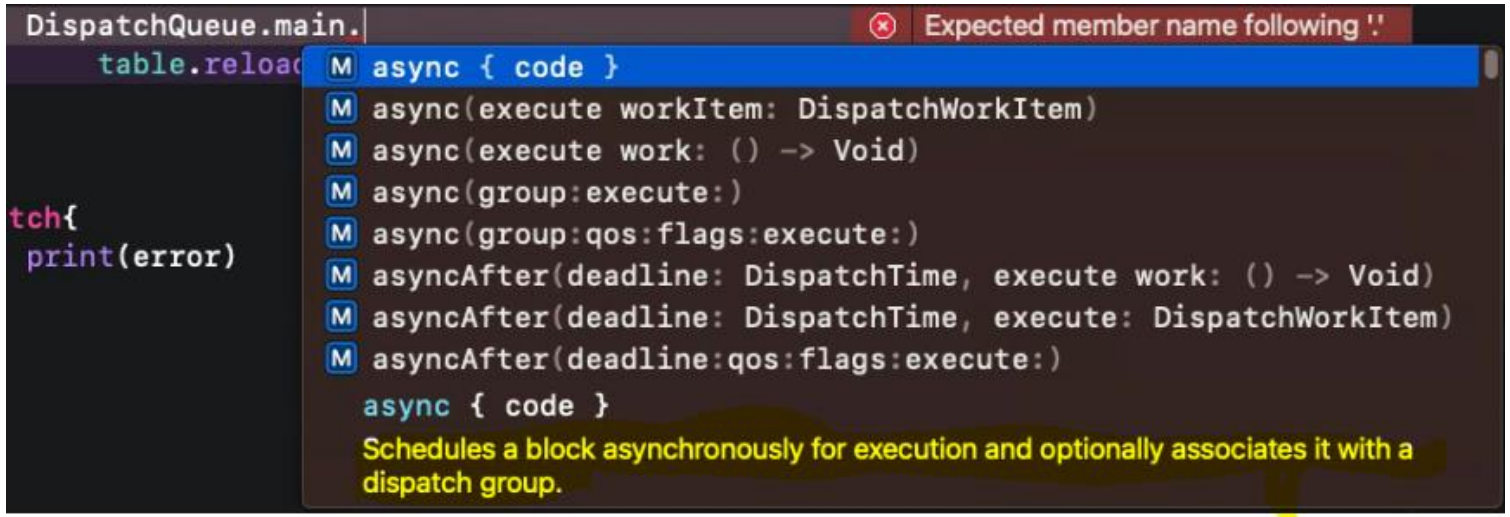
reloadData()

- https://developer.apple.com/documentation/uikit/uitableview/1614862-reloaddata
- 테이블 뷰의 row과 section을 다시 로드해 주어야죠!
- 그런데 main thread 관련오류 !

```
let decoder = JSONDecoder()
do{
    let decodedData = try decoder.decode(MovieData.self, from: JSONdata)
    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm)
    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].audiCnt)
    self.movieData = decodedData
    table.reloadData()          ⚠ UITableView.reloadData() must be used from main thread only

}catch{
    print(error)
}
```

UI관련 소스는 main thread에서 처리

- https://developer.apple.com/documentation/dispatch/dispatchqueue



앱의 기본 스레드 또는 백그라운드 스레드에서 작업 실행을 순차 또는 동시 처리로 관리하는 개체

현재 프로세스의 main 스레드와 관련된 디스패치 큐

```
DispatchQueue.main.async {
    self.table.reloadData()
}
```

비동기 처리

tableView(_:numberOfRowsInSection:) 수정

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return 10
}
```

결과


10:09
◀ MovieHsh
닥터 스트레인지: 대혼돈의 멀티버스
범죄도시 2
배드 가이즈
니 부모 얼굴이 보고 싶다
극장판 엉덩이 탐정: 수플레 섬의 비밀
신비한 동물들과 덤블도어의 비밀
민스미트 작전
엄마
서울괴담
어부바

guard문(조건식이 거짓이면 실행)

- guard문은 swift 2에 도입된 구문
- guard문은 표현식이 거짓(false)일 때 실행할 else 절을 반드시 포함해야 함
  - else 절에는 현재의 코드를 빠져 나갈 수 있는 구문(return, break, continue, throw 구문)을 반드시 포함해야 함

```
guard <불리언 표현식> else {
// 표현식이 거짓일 경우에 실행될 코드
<코드 블록을 빠져 나갈 구문>
}
// 표현식이 참일 경우에 실행되는 코드는 이곳에 위치
```

- guard문은 기본적으로 특정 조건에 맞지 않을 경우에 현재의 함수나 반복문에서 빠져 나갈 수 있도록 하는 '조기 출구(early exit)' 전략을 제공

guard~let의 활용

- guard는 return, break, continue, throw 등 제어문 전환 키워드를 쓸 수 있는 상황이라면 사용이 가능
- 그래서 함수 뿐 아니라 반복문 등 특정 블록 내부에 있으면 사용 가능
- 물론 함수 내부에 있다면 보통 return을 써서 해당 함수를 조기에 빠져나오는 조기 출구 용도로 사용
- 실제 앱을 만들다 보면 옵셔널 바인딩 때문에 다중 if~else를 사용해야 하는데, guard~let을 사용하면 다중 루프 없는 훨씬 가독성이 좋은 코드가 가능해서 그런 경우 많이 사용
- 다음은 while문 안에 guard와 break문을 사용한 경우
  - 이렇게 guard문을 활용하지는 않음

```
var x = 1
while true {
  guard x < 5 else { break }//조건(x<5)이 거짓일 때 실행(break)
  print(x) //1 2 3 4, 조건(x<5)이 참일 때 실행
  x = x + 1
}
```

```
var x = 1
while true {
  if x > 4 { break }
  print(x) //1 2 3 4
  x = x + 1
}
```

guard let~else로 옵셔널 바인딩

```
func multiplyByTen(value: Int?) {
  guard let number = value else {//조건식이 거짓(nil)일 때 else 블록 실행
     print("nil")
     return
  }
  print(number*10) //조건식이 참일 때 실행, 주의 : number를 guard문 밖인 여기서도 사용 가능
}
multiplyByTen(value: 3)   //30
multiplyByTen(value: nil)
multiplyByTen(value: 10)
```

- 언래핑된 number 변수를 guard문 밖에 있는 코드가 사용할 수 있다!!
  - if문을 이용하여 언래핑된 변수는 그렇게 못함

if~let vs. guard~let

```
func printName(firstName:String, lastName:String?){
// if let
if let lName = lastName {    // lastName이 nil이 아니면
  print(lName,firstName)
}
else {
  print("성이 없네요!")
}
// guard let
guard let lName = lastName else { // lastName이 nil이면
  print("성이 없네요!")
  return // early exit
}
print(lName,firstName)
}
printName(firstName: "길동", lastName:"홍")
printName(firstName: "길동", lastName:nil)
```

```
func printName(firstName:String, lastName:String?){
 if let lName = lastName {
   print(lName,firstName)
 }
 else {
   print("성이 없네요!")
 }
}
printName(firstName: "길동", lastName:"홍")
printName(firstName: "길동", lastName:nil)
```

```
func printName(firstName:String, lastName:String?){
 guard let lName = lastName else {
   print("성이 없네요!")
   return
 }
 print(lName,firstName)
}
printName(firstName: "길동", lastName:"홍")
printName(firstName: "길동", lastName:nil)
```

if let vs guard let : guard let이 소스 보기가 편해요

```swift
func getData(){
    if let url = URL(string:movieURL){
        let session = URLSession(configuration: .default)
        let task = session.dataTask(with: url) {(data, response, error) in
            if error != nil {
                print(error!)
                return
            }
            if let JSONdata = data{ //data가 옵셔널형임
                print(JSONdata,response!)
                let dataString = String(data: JSONdata, encoding: .utf8)
                print(dataString!) //데이터를 스트링으로 찍어보기
                let decoder = JSONDecoder()
                do{
                    let decodedData = try decoder.decode(MovieData.self, from
                    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].m
                    print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].a
                    self.movieData = decodedData
                    DispatchQueue.main.async {
                        self.table.reloadData()
                    }
                }catch{
                    print(error)
                }
            }
        }
        task.resume()
    }
}
```

```swift
func getData(){
    guard let url = URL(string: movieURL) else { return }
    //if let url = URL(string: movieURL) {
        let session = URLSession(configuration: .default)
        let task = session.dataTask(with: url) { [self] (data, response, error) in
            if error != nil {
                print(error!)
                return
            }
            guard let JSONdata = data else { return }
            //if let JSONdata = data {
            // print(JSONdata)
            let dataString = String(data: JSONdata, encoding: .utf8)
            print(dataString!)
            let decoder = JSONDecoder()
            do{
                let decodedData = try decoder.decode(MovieData.self, from: JSONdata)
                //print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm)
                //print(decodedData.boxOfficeResult.dailyBoxOfficeList[0].audiCnt)
                self.movieData = decodedData
                DispatchQueue.main.async {
                    self.table.reloadData()
                }

            }catch{
                print(error)
            }
            //  } //if let JSONdata = data {
        }
        task.resume()
        //}   //if let url = URL(string: movieURL) {
}
```

```
func getData(){ //1
  if let url = URL(string: movieURL) {  //2
    let session = URLSession(configuration: .default)
    let task = session.dataTask(with: url) { (data, response, error) in//3
      if error != nil {
          print(error!)
          return
      }
      if let JSONdata = data {  //4
        let dataString = String(data: JSONdata, encoding: .utf8)
        print(dataString!)
        let decoder = JSONDecoder()
        do{   //5
          let decodedData=try decoder.decode(MovieData.self,from:JSONdata)
          self.movieData = decodedData
            DispatchQueue.main.async {
               self.table.reloadData()
            }
        }catch{
             print(error)
         }  //5
        }  //4
      } //3
      task.resume()
   } //2
} //1
```

```
func getData(){ //1
  guard let url = URL(string: movieURL) else { return }
  let session = URLSession(configuration: .default)
  let task = session.dataTask(with: url) {(data, response, error) in //2
     if error != nil {
          print(error!)
          return
     }
     guard let JSONdata = data else { return }
     let dataString = String(data: JSONdata, encoding : .utf8)
     print(dataString!)
     let decoder = JSONDecoder()
     do{
       let decodedData = try decoder.decode(MovieData.self, from: JSONdata)
       self.movieData = decodedData
         DispatchQueue.main.async{
           self.table.reloadData()
         }
      }catch {
         print(error)
      }
    } //2
    task.resume()
} //1
```

[Editor]-[Structure]-[Re-Indent]

ViewController 클래스 소스가 너무 복잡해요!

- 소스를 몇 개로 나누면 더 깔끔하겠죠.

```
8  class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
◉      @IBOutlet weak var table: UITableView!
10     let movieURL =
           "https://kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/searchDailyBoxOfficeList
           .json?key=[redacted]515"
11
12     override func viewDidLoad() {
13         super.viewDidLoad()
14         // Do any additional setup after loading the view.
15         table.delegate = self
16         table.dataSource = self
17         getData()
18     }
```

과제 : 변형

```swift
import UIKit
let name = ["aaa", "bbb","ccc","ddd","eee"]
struct MovieData : Codable {
    let boxOfficeResult : BoxOfficeResult
}
struct BoxOfficeResult : Codable {
    let dailyBoxOfficeList : [DailyBoxOfficeList]
}
struct DailyBoxOfficeList : Codable {
    let movieNm : String
    let audiCnt : String
}
class ViewController: UIViewController, UITableViewDelegate, UITabl
                                    table: UITableView!
```

1 2 3

주의:Codable
프로토콜
준수해야 함

```
▼ object {1}
▼ boxOfficeResult {3}
    boxofficeType : 일별 박스오피스
    showRange : 20220514~20220514
▼ dailyBoxOfficeList [10]
  ▼ 0 {18}
      rnum : 1
      rank : 1
      rankInten : 0
      rankOldAndNew : OLD
      movieCd : 20212855
      movieNm : 닥터 스트레인지:
```

Summary

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.

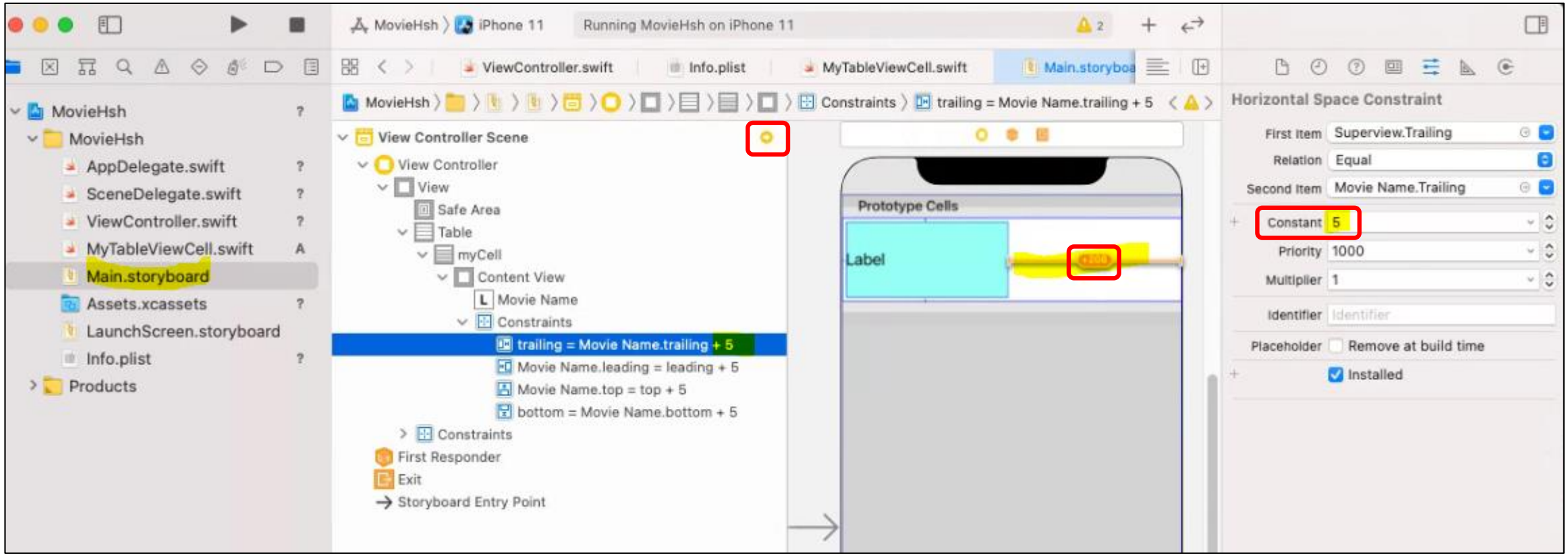

```
{
  "boxOfficeResult": {
    "boxofficeType": "일별 박스오피스",
    "showRange": "20220514~20220514",
    "dailyBoxOfficeList": [
      {
        "rnum": "1",
        "rank": "1",
        "rankInten": "0",
        "rankOldAndNew": "OLD",
        "movieCd": "20212855",
        "movieNm": "닥터 스트레인지:
        "openDt": "2022-05-04",
        "salesAmt": "4050678820",
        "salesShare": "72.3",
        "salesInten": "2127489900",
        "salesChange": "110.6",
        "salesAcc": "49191168260",
        "audiCnt": "369436",
        "audiInten": "193888",
        "audiChange": "110.4",
        "audiAcc": "4612609",
        "scrnCnt": "2495",
        "showCnt": "11403"
      },
      {
        "rnum": "2",
        "rank": "2",
        "rankInten": "0",
        "rankOldAndNew": "NEW",
        "movieCd": "20204548",
        "movieNm": "범죄도시 2",
        "openDt": "2022-05-18",
        "salesAmt": "794804440",
        "salesShare": "14.2",
        "salesInten": "794804440",
        "salesChange": "100",
```


응답 구조

응답 필드	값	
boxofficeType	문자열	박스오피스 종류를 출력합니다
showRange	문자열	박스오피스 조회 일자를 출력
rnum	문자열	순번을 출력합니다.
rank	문자열	해당일자의 박스오피스 순위를
rankInten	문자열	전일대비 순위의 증감분을 출력
rankOldAndNew	문자열	랭킹에 신규진입여부를 출력합 "OLD" : 기존 , "NEW" : 신규
movieCd	문자열	영화의 대표코드를 출력합니다
movieNm	문자열	영화명(국문)을 출력합니다.
openDt	문자열	영화의 개봉일을 출력합니다.
salesAmt	문자열	해당일의 매출액을 출력합니다
salesShare	문자열	해당일자 상영작의 매출총액 대
salesInten	문자열	전일 대비 매출액 증감분을 출
salesChange	문자열	전일 대비 매출액 증감 비율을
salesAcc	문자열	누적매출액을 출력합니다.
audiCnt	문자열	해당일의 관객수를 출력합니다
audiInten	문자열	전일 대비 관객수 증감분을 출
audiChange	문자열	전일 대비 관객수 증감 비율을
audiAcc	문자열	누적관객수를 출력합니다.
scrnCnt	문자열	해당일자에 상영한 스크린수를
showCnt	문자열	해당일자에 상영된 횟수를 출력

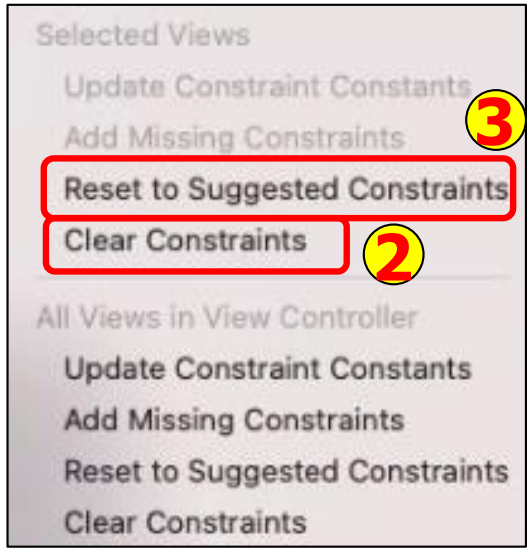
```
decodedData.boxOfficeResult.dailyBoxOfficeList[0].movieNm
```

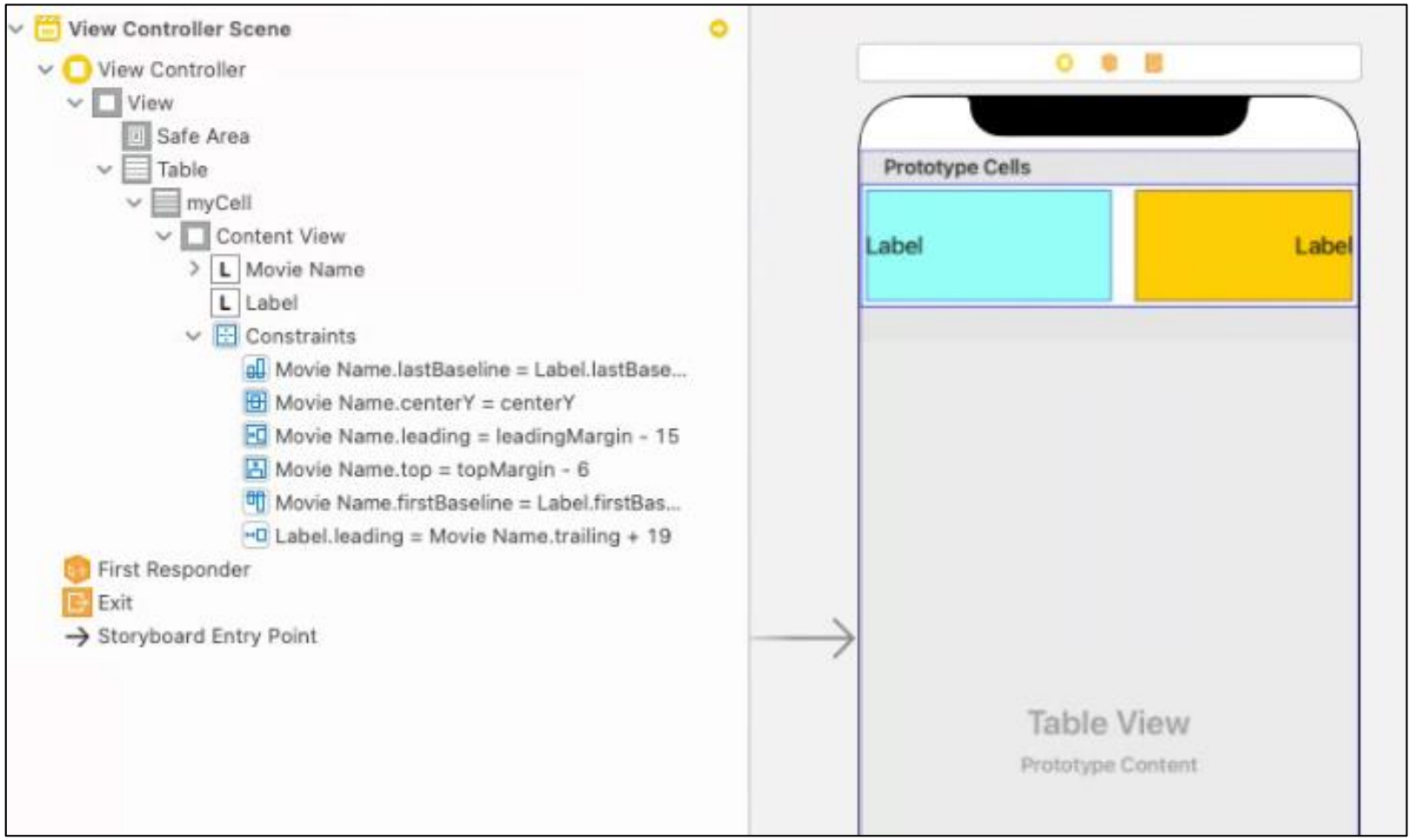
Label 디자인 변경시 : Resolve Auto Layout Issues



Selected Views vs All Views in View Controller

- 가장 쉬운 방법: Label 모두 선택한 후 Clear후 Reset





개선 : 앱을 실행하면 어제 날짜로 자동 조회하기

```
19  class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
    @IBOutlet weak var table: UITableView!
21      var movieData : MovieData?
22      var movieURL =
            "https://kobis.or.kr/kobisopenapi/webservice/rest/boxoffice/searchDailyBoxOfficeList
            .json?key=[redacted]&targetDt=" //20220514
23
24      override func viewDidLoad() {
25          super.viewDidLoad()
26          // Do any additional setup after loading the view.
27          table.delegate = self
28          table.dataSource = self
29
30          movieURL +=  makeYesterdayString()
31          getData()
32      }
33      func makeYesterdayString() -> String {
34          let y = Calendar.current.date(byAdding: .day, value: -1, to: Date())!
35          let dateF = DateFormatter()
36          dateF.dateFormat = "yyyyMMdd"
37          let day = dateF.string(from: y)
38          return day
39      }
40      func getData(){
41          if let url = URL(string:movieURL){
```

주의 : 여기는 날짜 삭제

날짜 없는 movieURL 뒤에 20230509을 붙여줌

오늘이 2023년 5월 10일이면 어제 날짜를 20230509 형식으로 만들어야 해요.
Calendar swift, DateFomatter swift 등으로 구글링