에러처리

강사 주영민

예외처리

- 프로그램의 오류 조건에 응답하고 오류 조건에서 복구하는 프로세스입니다

- Swift는 런타임시 복구 가능한 오류를 던지고, 포착하고, 전파하고, 조작하는 기능을 제공합니다.

- 에러는 Error 프로토콜을 준수하는 유형의 값으로 나타냅니다. 실제로 Error프로토콜은 비어 있으나 오류를 처리할수 있는 타입임을 나타냅니다.

열거형의 에러 표현

- 열거형은 에러를 표현하는데 적합합니다.

```
enum VendingMachineError: Error {
    case invalidSelection
    case insufficientFunds(coinsNeeded: Int)
    case outOfStock
}
```

에러발생

- throw 키워드를 통해 에러를 발생 시킵니다.

```
throw VendingMachineError.insufficientFunds(coinsNeeded: 5)
```

에러전달

- 함수의 작성중 에러가 발생할수 있는 함수에는 매개변수 뒤에 throws 키워드를 작성하여 에러가 전달될수 있는 함수를 선언합니다.

```
//에러전달 가능성 함수
func canThrowErrors() throws -> String

//에러전달 가능성이 없는 함수
func cannotThrowErrors() -> String
```

에러처리

- 함수가 에러를 throw하면 프로그램의 흐름이 변경되므로 에러가 발생할 수있는 코드의 위치를 신속하게 식별 할 수 있어야합니다.

- 이 장소를 식별하기 위해 try 나 try?, try!를 사용할수 있습니다.

- 발결된 에러를 처리하기 위해 do-catch 문을 사용해서 에러를 처리 합니다.

do - catch

```
do {
    try expression
    statements
} catch pattern 1 {
    statements
} catch pattern 2 where condition {
    statements
}
```

예제

```
let favoriteSnacks = [
    "Alice": "Chips",
    "Bob": "Licorice",
    "Eve": "Pretzels",
]


func buyFavoriteSnack(person: String, vendingMachine: VendingMachine) throws {
    let snackName = favoriteSnacks[person] ?? "Candy Bar"
    try vendingMachine.vend(itemNamed: snackName)
}
```

```
var vendingMachine = VendingMachine()
vendingMachine.coinsDeposited = 8

do {
    try buyFavoriteSnack(person: "Alice", vendingMachine: vendingMachine)
} catch VendingMachineError.invalidSelection {
    print("Invalid Selection.")
} catch VendingMachineError.outOfStock {
    print("Out of Stock.")
} catch VendingMachineError.insufficientFunds(let coinsNeeded) {
    print("Insufficient funds. Please insert an additional \(coinsNeeded) coins.")
}
```

예제

```
class VendingMachine {
    var inventory = [
        "Candy Bar": Item(price: 12, count: 7),
        "Chips": Item(price: 10, count: 4),
        "Pretzels": Item(price: 7, count: 11)
    ]
    var coinsDeposited = 0

    func vend(itemNamed name: String) throws {
        guard let item = inventory[name] else {
            throw VendingMachineError.invalidSelection
        }
        guard item.count > 0 else {
            throw VendingMachineError.outOfStock
        }

        guard item.price <= coinsDeposited else {
            throw VendingMachineError.insufficientFunds(coinsNeeded: item.price - coinsDeposited)
        }

        coinsDeposited -= item.price

        var newItem = item
        newItem.count -= 1
        inventory[name] = newItem

        print("Dispensing \(name)")
    }
}
```

```
struct Item {
    var price: Int
    var count: Int
}

enum VendingMachineError: Error {
    case invalidSelection
    case
insufficientFunds(coinsNeeded: Int)
    case outOfStock
}
```

Converting Errors to Optional Value

```swift
func someThrowingFunction() throws -> Int {
    // ...
}

let x = try? someThrowingFunction()

let y: Int?
do {
    y = try someThrowingFunction()
} catch {
    y = nil
}
```

Specifying Cleanup Actions (후처리)

- 에러에 의해 함수의 문제가 생기더라도 꼭! 해야할 행동이 있다면!!
- defer 구문은 블럭이 어떻게 종료되던 꼭 실행된다는 것을 보장.

```
func processFile(filename: String) throws {
    if exists(filename) {
        let file = open(filename)
        defer {
            close(file)
        }

        while let line = try file.readline() {
            // Work with the file.
        }
        // close(file) is called here, at the end of the scope.
    }
}
```

Data 저장

강사 주영민

Property list

Key	Type	Value
▼ Information Property List	Dictionary	(14 items)
Localization native development re...	String	en
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone enviro...	Boolean	YES
Launch screen interface file base...	String	LaunchScreen
Main storyboard file base name	String	Main
▼ Required device capabilities	Array	(1 item)
Item 0	String	armv7
▶ Supported interface orientati...	Array	(3 items)

Property list - plist

- 심플한 “파일” 저장 방법 중 하나.
- Key,Value구조로 데이터 저장
- File 형태로 저장되다 보니 외부에서 Access가능(보안 취약)

파일 위치

- 파일이 저장되는곳 Bundle & Documents 폴더
- Bundle은 프로젝트에 추가된 Resorce가 모인 곳
- 프로그램이 실행되며 저장하는 파일은 Documents폴더에 저장된다.
- **즉! plist파일의 데이터만 불러오는 역할은 Bundle을 통해서, plist파일에 데이터를 쓰고 불러오는 역할은 Documents폴더에 저장된 파일로!**

Plist File In Bundle

1. bundle에 있는 파일 Path 가져오기
2. Path를 통해 객체로 변환, 데이터 불러오기
3. 사용

Bundle

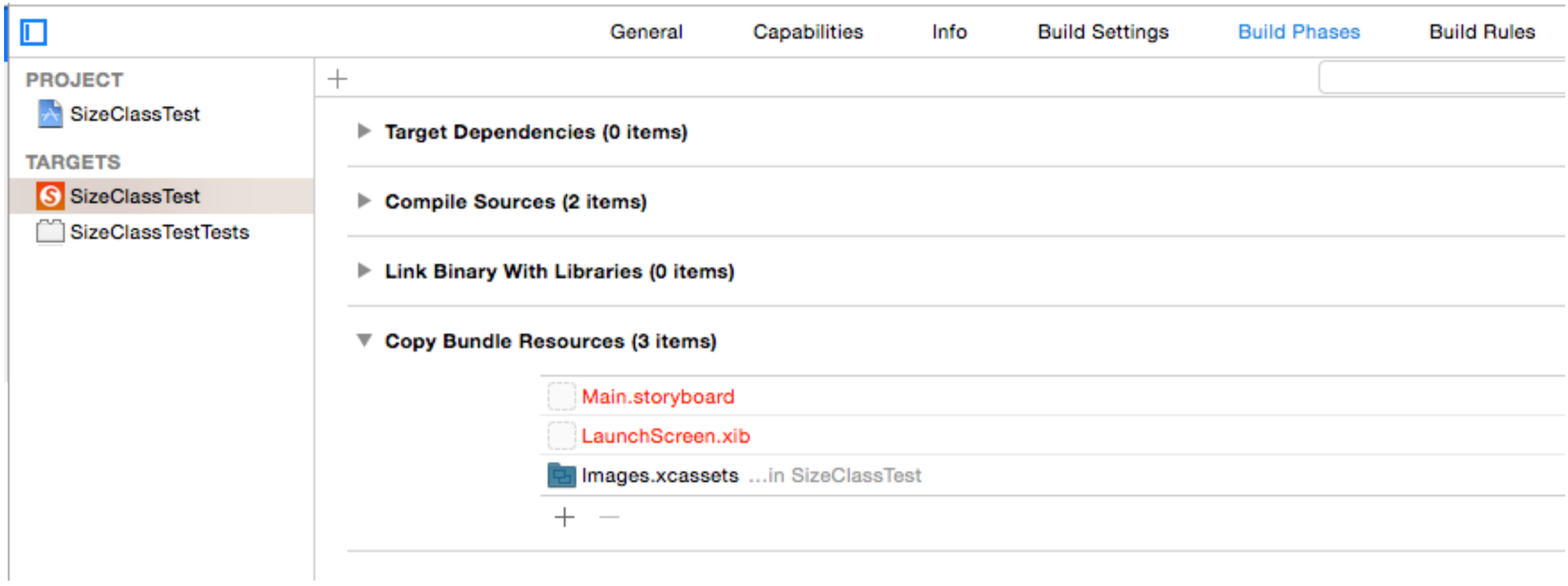
강사 주영민

Fast campus

Bundle

- 실행코드, 이미지, 사운드, nib 파일, 프레임 워크,설정파일 등 코드와 리소스가 모여있는 file system내의 Directory

Bundle 확인



Main Bundle 가져오기

```swift
// Get the main bundle for the app.
let mainBundle = Bundle.main
```

Bundle 파일 주소 가져오기

```
// Get the main bundle for the app.
let mainBundle = Bundle.main
let filePaht = mainBundle.path(forResource: "rName", ofType:
"rType")
```

데이터 불러오기

```
if let path = filePaht {

    let image = UIImage(contentsOfFile: filePaht!)

}
```

Plist File In Bundle

```swift
if let filePaht = mainBundle.path(forResource: "rName", ofType: "rType"),
   let dict = NSDictionary(contentsOfFile: filePaht) as? [String: AnyObject]
{
   // use swift dictionary as normal
}
```

Plist File In Document

1. Document folder Path 찾기

2. Document folder에 plist 파일이 있는지 확인
   - 만약 없다면 : bundle에 있는 파일 Document에 복사

3. Path를 통해 객체로 변환, 데이터 불러오기

4. writeToFile 메소드로 파일 저장

1. 파일 불러오기 (NSSearchPathForDirectoriesInDomains)

```
let path:[String] =
NSSearchPathForDirectoriesInDomains(.documentDirectory, .userDomainMask,
true)
let basePath = path[0] + "/fileName.plist"
```

- doucument 폴더에 Path구하기

2. Document folder에 파일 있는지 확인

```
if !FileManager.default.fileExists(atPath: basePath)
{

}
```

- doucument 폴더에 plist파일이 존재 하지는지 확인

3. bundle에 있는 파일 Document에 복사

```
 if let fileUrl = Bundle.main.path(forResource: "fileName", ofType: "plist")
{
    do {
        try FileManager.default.copyItem(atPath: fileUrl, toPath: basePath)
    } catch  {
        print("fail")
    }
}
```

- 만약 doucument에 해당 plist파일이 존재 하지 않을때, bundle에 있는 파일을 document폴더로 복사

4. Dictionary 인스턴스 불러오기

```swift
if let dict = NSDictionary(contentsOfFile: basePath) as? [String: AnyObject]
{
   // use swift dictionary as normal
}
```

- document 폴더에 있는 파일을 NSDictionary을 통해서 Dictionary인스턴스로 불러오기

5. write(toFile)메소드를 통해 파일 저장

```
if let dict = NSDictionary(contentsOfFile: basePath) as? [String: Any]
{
    var loadData = dict
    loadData.updateValue("addData", forKey: "key")

    let nsDic:NSDictionary =  NSDictionary(dictionary: loadData)
    nsDic.write(toFile: basePath, atomically: true)
}
```

- dictionary를 수정
- NSDictionary로 변경후 writeTofile 메소드를 통해 파일에 저장

실습

- 한번 같이 만들어 볼까요?

Singletone

강사 주영민

Singleton Pattern

1. 싱글톤 이란? 어플리케이션 전 영역의 걸쳐 하나의 클래스의 단 하나의 인스턴스만(객체)을 생성하는 것을 싱글톤 패턴이라고 한다.

2. 사용이유 : 어플리케이션 내부에서 유일하게 하나만 필요한 객체에서 사용(셋팅, 데이터 등)

3. 클래스 메소드로 객체를 만들며 static을 이용해서 단 1개의 인스턴스만 만든다.

4. 앱 내에서 공유하는 객체를 만들 수 있다.

Singleton Pattern 인스턴스 만들기

```
class SingletonClass {
    // MARK: Shared Instance
    static var sharedInstance:SingletonClass = SingletonClass()

    // Can't init is singleton
    private init()
    {
        //초기화가 필요하면 private로 생성
    }

}
```

Singleton Pattern 예제

```
//스크린 정보를 가지고 있는 객체
let screen = UIScreen.main

//사용자 정보를 저장하는 객체
let data = UserDefaults.standard

//어플리케이션 객체
let app = UIApplication.shared

//파일 시스템 정보를 가지고 있는 객에
let fileManager = FileManager.default
```

“싱글턴 패턴은 왜 사용하는 것일까?”