

Class Project: Applied Delegation

METCS-665 Spring-02

Christopher W Gonzalez Melendez - April 29th, 2021

Introduction

Delegation Software Design Pattern

What is Delegation design pattern

A Business Case

Delegation Software Design Pattern

- Resolves bloated Controllers or objects with behaviors
- Contributes to resolves bloated technical documentation
- Applies modularization technique to the architecture
- Delegate objects can be exchanged and/or shared in form of frameworks
- Promotes decoupling

How It Works

Delegation Software Design Pattern

```
1  //
2  // ProfilesTableViewControllerPresenter.swift
3  // SwiftDelegation
4  //
5  // Created by Christopher Gonzalez on 4/29/21.
6  //
7
8  import UIKit
9
10 /**
11  DELEGATION COMPONENT: The delegate/presenter object
12
13  A class for handling the delegated behaviour for the presentation of the `ProfileTableViewController`
14  The `UIAdaptivePresentationControllerDelegate` is the interface which declares the behavior we are delegating to this object.
15  This object implements the behavior conforming with the needs of the controller that is delegating the behavior.
16  */
17 class ProfilesTableViewControllerPresenter: NSObject, UIAdaptivePresentationControllerDelegate {
18
19     // A native delegate method that will execute upon the user swipes down to dismiss
20     func presentationControllerWillDismiss(_ presentationController: UIPresentationController) {
21
22         if let profilesController = presentationController.delegate as? ProfilesTableViewController {
23
24             let dataArray = UserDefaults.standard.array(forKey: ProfilesTableViewController.keyId) as? [Data] ?? [Data]()
25
26             profilesController.dataSource.profiles = dataArray
27             profilesController.tableView.reloadData()
28         }
29     }
30 }
31
```

How It Works

Delegation Software Design Pattern

```
6 //
7
8 import UIKit
9
10 /**
11  DELEGATION COMPONENT: The delegate/data-source/presenter association holder in form of a controller
12
13  A class for managing and controlling the views and data of the table list.
14  This object hold the reference of the delegate/data-source and also holds the view object that relies on the behavior for presenting the data.
15  The binding of the delegate is made through a storyboard object and referenced/associated here as an outlet binder.
16  */
17 class ProfilesTableViewController: UITableViewController {
18
19     // The key id for storing the data
20     public static let keyId = "edu.bu.gchriswgm.SwiftDelegation.UserDefaults.profiles.KEY"
21
22     /**
23      DELEGATION COMPONENT: The delegate/data-source associated object.
24
25      This property holds the association/reference in form of a property outlet that is linked or binded via the storyboard.
26      We can held it as strong reference due to the fact that ARC will manage and release the reference.
27      */
28     @IBOutlet var dataSource: ProfileTableViewDataSource!
29
30     /**
31      DELEGATION COMPONENT: The delegate/data-source associated object.
32
33      This property holds the association/reference in form of a property outlet that is linked or binded via the storyboard.
34      We can held it as strong reference due to the fact that ARC will manage and release the reference.
35      */
36     @IBOutlet var presenter: ProfilesTableViewControllerPresenter!
37
```

Implementation Overview


Delegation Software Design Pattern

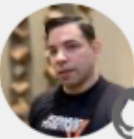

Code Deep Dive



Unit Test Overview

Delegation Software Design Pattern

Travis CI


 Dashboard Changelog Documentation Help



 gchriswill / METCS665-FINAL 



build passing



Current Branches Build History Pull Requests


More options 


✓ feature/delegation Adds 3 UMLs for the applied delegation pattern


 Commit 85d628a 

 Compare f0ccca2..85d628a 

 Branch feature/delegation 


 Christopher Gonzalez


 Ran for 6 min 26 sec

 20 minutes ago

Restart build

Debug build

 </> Xcode: xcode12.2 Xcode scheme: SwiftDelegation

 AMD64

Unit Test Overview

Delegation Software Design Pattern

```
835 All tests
836 Test Suite SwiftDelegationTests.xctest started
837 SwiftDelegationTests
838   ✓ testClearAll (0.073 seconds)
839   ✓ testCreateProfile (0.011 seconds)
840   ✓ testDeleteProfile (0.011 seconds)
841   ✓ testUpdateProfile (0.018 seconds)
842
843
844   Executed 4 tests, with 0 failures (0 unexpected) in 0.114 (0.122) seconds
845
846 2021-04-29 09:22:50.907 xcodebuild[2532:8015] [MT] IDETestOperationsObserverDebug: 314.522 elapsed — Testing started completed.
847 2021-04-29 09:22:50.908 xcodebuild[2532:8015] [MT] IDETestOperationsObserverDebug: 0.000 sec, +0.000 sec — start
848 2021-04-29 09:22:50.908 xcodebuild[2532:8015] [MT] IDETestOperationsObserverDebug: 314.522 sec, +314.522 sec — end
849 ▶ Test Succeeded
850 The command "set -o pipefail && xcodebuild -project SwiftDelegation/SwiftDelegation.xcodeproj -scheme SwiftDelegation -destination platform=iOS
12,OS=14.2 build test | xcpretty" exited with 0.
851
852
853 Done. Your build exited with 0.
```


How to Further Improve Delegation Software Design Pattern

A way this pattern could be improved is by a technique I named “Delegate Forwarding”, which is basically grouping the behaviors already in a delegate in categories and further delegating the categories to new associated delegate object.