iOS alapú szoftverfejlesztés - Labor 12

A laborsegédletet összeállította: Blázovics László, Kántor Tibor és Kelényi Imre

A labor témája:

* + Third-party Swift komponens integrálása
  + Objective-C kód integrálása
  + A Cocoapods használata

A laborhoz tartozó nagyobb kódrészletek a következő url-en érhetők el "copy-paste barát" formában:

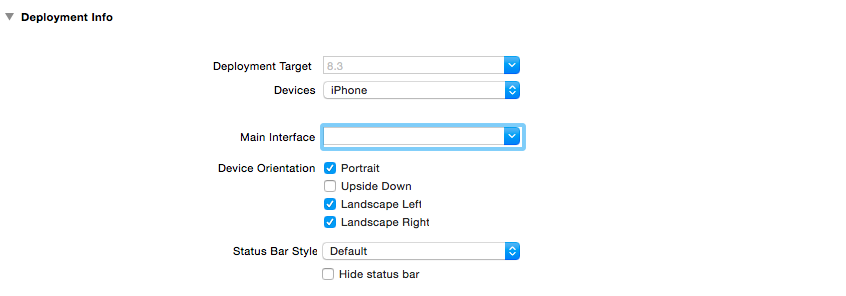
https://gist.github.com/blazovics/416a6c5d431ffdd161a9

# ImageDownloader

Alkalmazásunk a fortepan.hu-ról fog letölteni közepes, illetve nagy felbontású képeket. Utóbbi esetén azt is szeretnénk megmutatni a felhasználónak, hogy hogyan halad a letöltés.

Kiindulásnak hozzunk lére egy SingleView applicationt **PictureDownload** néven.

A létrehozott projektben töröljük ki a Main.Storyboard-ot és ezt vezessük át project beállításaiban is, nevezetesen töröljük ki a Main interface mező tartalmát a Deployment Info résznél.



## ViewController inicializálása kódból

Adjuk hozzá az alábbi kódrészletet az AppDelegate didFinishLaunchingWithOptions metódusához:

func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {

self.window = UIWindow(frame: UIScreen.mainScreen().bounds)

let viewController = ViewController()

viewController.imageUrl =

NSURL(string: "http://www.fortepan.hu/\_photo/display/28268.jpg")

viewController.contentUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/download/fortepan\_28268.jpg")

self.window!.rootViewController = viewController

self.window!.makeKeyAndVisible()

return true

}

https://gist.github.com/blazovics/416a6c5d431ffdd161a9#file-appdelegate-didfinishlaunchingwithoptions-swift

Ezután menjünk át a ViewController.swift-re és kezdjük el kiegészíteni.

Hozzunk létre egy **imageView** nevű UIImageView property-t, hogy szükség esetén le tudjuk cserélni a benne lévő képet, illetve egy **contentUrl** és egy **imageUrl** property-t

class ViewController: UIViewController {

var contentUrl:NSURL?

var imageUrl:NSURL?

var imageView:UIImageView?

Majd cseréljük le a viewDidLoad metódust a következő kódrészlettel:

override func viewDidLoad() {

super.viewDidLoad()

imageView = UIImageView(frame: CGRectMake(self.view.bounds.origin.x,

self.view.bounds.origin.y, self.view.bounds.size.width,

self.view.bounds.size.height/2))

imageView!.contentMode = UIViewContentMode.ScaleAspectFit

imageView!.backgroundColor = UIColor.blackColor()

self.view.addSubview(imageView!)

let downloadButton = UIButton(type: .System)

downloadButton.frame = CGRectMake(0, 0, 100, 40)

downloadButton.center = CGPointMake(self.view.bounds.width/2,

self.view.bounds.size.height\*5/8)

downloadButton.setTitle("Download", forState: UIControlState.Normal)

downloadButton.setTitleColor(

UIColor.blackColor(), forState: UIControlState.Normal)

downloadButton.addTarget(

self, action: "downloadFile",

forControlEvents: UIControlEvents.TouchUpInside)

self.view.addSubview(downloadButton)

let displayButton = UIButton(type: .System)

displayButton.frame = CGRectMake(0, 0, 100, 40)

displayButton.center =

CGPointMake(self.view.bounds.width/2, self.view.bounds.size.height\*6/8)

displayButton.setTitle("Present", forState: UIControlState.Normal)

displayButton.setTitleColor(

UIColor.blackColor(), forState: UIControlState.Normal)

self.view.backgroundColor = UIColor.whiteColor()

self.view.addSubview(displayButton)

}

https://gist.github.com/blazovics/416a6c5d431ffdd161a9#file-viewcontroller-viewdidload-swift

Teszteljük az alkalmazást!

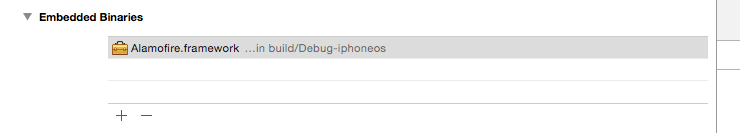
Látható, hogy minden UI elem megjelenik, de a kép nem töltődik be, és a letöltés gomb megnyomásakor is hibát kapunk.

A kép letöltéséhez használjuk az Alamofire nevű third party komponenst a következő linkről:

<https://github.com/Alamofire/Alamofire/archive/master.zip>

Tömörítsük ki az állományt, majd az Alamofire.xcodeproj fájlt adjuk hozzá a projekthez. Legegyszerűbben ezt úgy tudjuk megtenni, hogy Finderből áthúzzuk az Xcode-ra.

Ezután adjuk hozzá a Embedded Binary-ként a megfelelő. Alamofire.framework-öt. Ehhez menjünk el a projektbeállításokhoz és a megfelelő helyen nyomjuk meg a + gombot, majd válasszük ki az iOS-re generált frameworköt.



Ezután nincs más dolgunk, mint, hogy a ViewControllerben importoljuk az Alamofire-t és felülírjuk a viewWillAppear metódust.

import UIKit

import Alamofire

override func viewWillAppear(animated: Bool) {

if imageUrl != nil{

Alamofire.request(.GET, imageUrl!)

.response(){ (\_, \_, data, \_) in

if let image = UIImage(data: data!) {

dispatch\_async(dispatch\_get\_main\_queue(),{ \_ in

self.imageView?.image = image

})

}

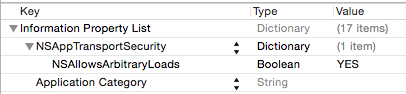
}

}

}

https://gist.github.com/blazovics/416a6c5d431ffdd161a9#file-viewcontroller-viewwillappear-swift

Ahogy már a korábbi laboroknál is előjött, engedélyezni kell az projekthez tartozó Info.plist fájlban a HTTP felett történő kommunikációt a következő két kulcs felvételével: NSAppTransportSecurity (Dictionary) és ezen belül az NSAllowsArbitraryLoads (BOOL).



Ha nem sikerülne, akkor a következő hibaüzenetet kapjuk:

**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.**

Most, hogy letöltődik a közepes méretű kép, töltsük le a nagyobb képet is. Ehhez először implementáluk az NSURLConnection-höz szükséges delegateket és további három property-t, amik a letöltött nagy képet , illetve a letöltés állapotát fogják nyilvántartani.

class ViewController: UIViewController, NSURLConnectionDelegate, NSURLConnectionDataDelegate{

var data: NSMutableData?

var contentLength: Int64 = 0

var downloadedContentLength: Int64 = 0

var contentUrl:NSURL?

var imageUrl:NSURL?

var imageView:UIImageView?

Ezután implementáljuk a downloadFile() nevű függvényt, amit a Download gomb eseménykezelője hív meg és az alkalmazás Documents mappájába menti a képet:

func downloadFile(){

let destination =

Alamofire.Request.suggestedDownloadDestination(

directory: .DocumentDirectory, domain: .UserDomainMask)

Alamofire.download(.GET, contentUrl!, destination: destination)

.progress { bytesRead, totalBytesRead,

totalBytesExpectedToRead in

print(totalBytesRead)

dispatch\_async(dispatch\_get\_main\_queue()) {

print("\(totalBytesRead)")

}

}

.response { \_, \_, \_, error in

if let error = error {

print("error: \(error)")

} else {

print("success")

}

}

}

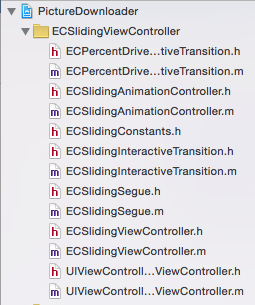
## ECSlidingViewController

Most, hogy van már rendes tartalmunk adjuk hozzá az alkalmazáshoz egy menüt.

Ehhez töltsük le az ECSlidingViewController thrid-party komponenst a következő linkről:

<https://github.com/ECSlidingViewController/ECSlidingViewController/archive/master.zip>

Csomagoljuk ki a fájlt és keressük meg az ECSlidingViewController mappát, Az egész mappát adjuk hozzá a projekthez.



Először hozzunk létre egy MenuTableViewController nevű UITableViewControllerből leszármazott osztályt, mely megvalósítja a showMenu metódust. Erre majd a navigáció során lesz szükségünk, hogy az unwingSegue-t be tudjuk kötni.

import UIKit

class MenuTableViewController: UITableViewController {

@IBAction func showMenu(seque:UIStoryboardSegue){

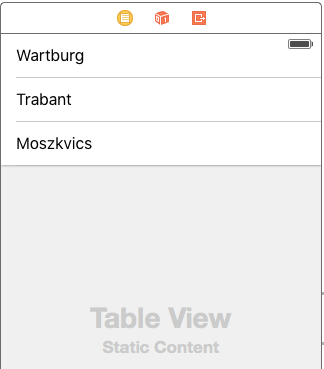
}

}

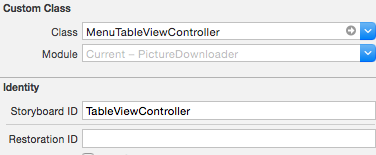
Nyissuk meg a storyboardot és adjunk hozzá egy TableViewControllert.

Állítsuk be a Custom Class szekcióban, hogy MenuTableViewController legyen.

A TableView-t kiválasztva állítsuk át a Content-et Static Cellsre, majd adjunk hozzá 3 cellát ahogy a képen is látható!



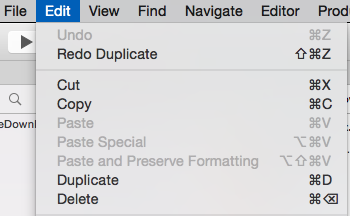
Állítsuk be a MenuTableViewController StoryboardID-jét TableViewController-re!



Ezután adjunk hozzá egy újabb ViewControllert, ami ECSlidingViewController legyen. Állítsuk be, hogy ez a komponens legyen a kezdő, azaz Initial View Controller!

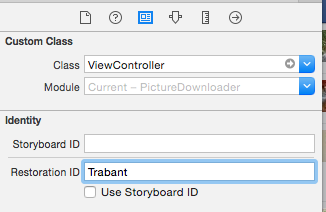
Ágyazzuk be a ViewControllert egy NavigationControllerbe, majd adjunk hozzá egy BarButtonItem-et Menu felirattal.

Duplikáljuk kétszer egymás után a ViewControlert és a NavigationControllert az Edit/Duplicate funkció haszánlatával.

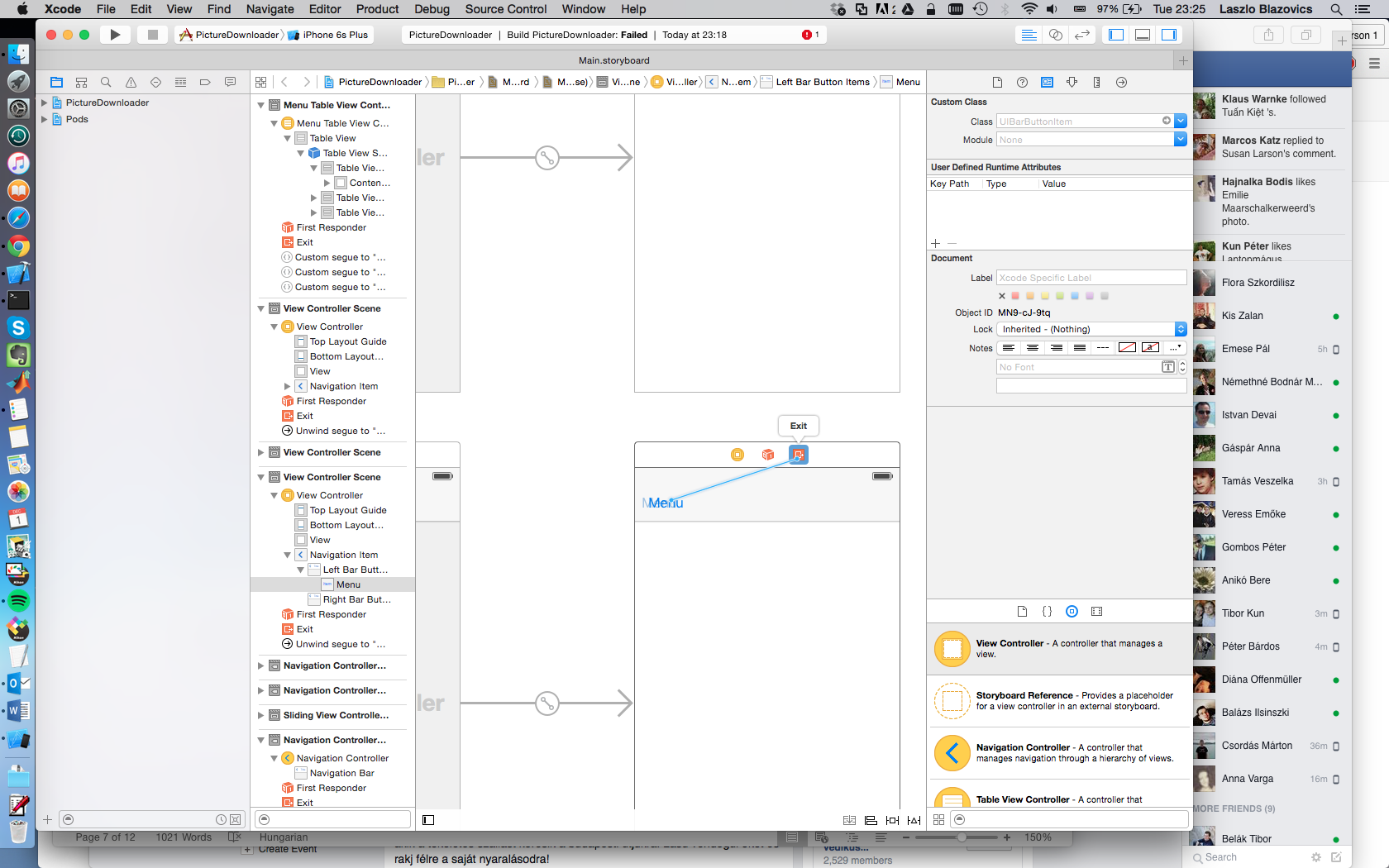


Az így létrejött 3 NavigationControllert kössük be a TableView egyes celláihoz egy-egy sliding segue-el.

A 3 ViewControllerben állítsuk be a RestorationID-ket a celláknak megfelelően. Ha a menüben Trabant volt a cella szövege, akkor az ahhoz tartozó ViewControllerben is legyen Trabant a RestorationID!

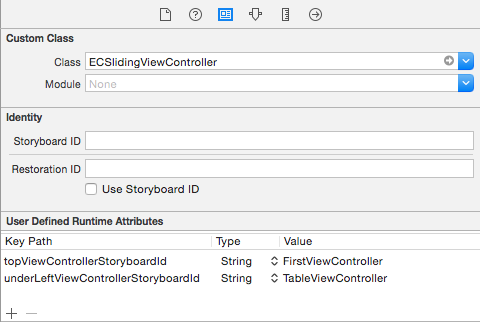


Ezután állítsuk be az Unwind segue-eket a három ViewControllerben a Menü gombhoz.



A Wartburghoz tartozó NavigationController StoryboardID-jét állítsuk FirstViewController-re.

Végül térjünk vissza a ECSlidingViewController-hez és adjunk hozzá két String típusú User Defined Runtime Attribute-ot (topViewControllerStoryboardId, underLeftViewControllerStoryboardId) a megfelelő értékekkel :



Már van menünk, de a többi nézetekhez nincs képünk. Ezt orvosolandó, egészítsük ki a ViewController viewDidLoad metódusát!

override func viewDidLoad() {

super.viewDidLoad()

let restorationID = self.restorationIdentifier!

switch restorationID {

case "Wartburg":

self.view.backgroundColor =

UIColor(red:0.13, green:0.14, blue:0.15, alpha:1)

self.imageUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/display/28268.jpg")

self.contentUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/download/fortepan\_28268.jpg")

case "Trabant":

self.view.backgroundColor =

UIColor(red:0.95, green:0.95, blue:0.95, alpha:1)

self.imageUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/display/74394.jpg")

self.contentUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/download/fortepan\_74394.jpg")

case "Moszkvics":

self.view.backgroundColor =

UIColor(red:0.8, green:0, blue:0.48, alpha:1)

self.imageUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/display/16022.jpg")

self.contentUrl =

NSURL(string:

"http://www.fortepan.hu/\_photo/download/fortepan\_16022.jpg")

default:

break

}

imageView = UIImageView(frame:

CGRectMake(self.view.bounds.origin.x, self.view.bounds.origin.y, self.view.bounds.size.width, self.view.bounds.size.height/2))

imageView!.contentMode = UIViewContentMode.ScaleAspectFit

imageView!.backgroundColor = UIColor.blackColor()

https://gist.github.com/blazovics/416a6c5d431ffdd161a9#file-menuviewcontroller-viewcontrollerforindexpath-swift

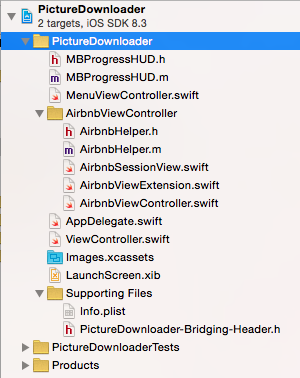
## MBProgressHUD

Most, hogy már van mit letölteni adjuk hozzá a projekthez az MBProgressHUD third-party plugint, amivel különböző progress bar-okat tudunk megjeleníteni.

Ehhez először töltsük le a forrást majd csomagoljuk is ki.

<https://github.com/jdg/MBProgressHUD/archive/master.zip>

Keressük meg és adjuk hozzá a projekthez az MBProgressHUD.h és az MBProgressHUD.m fájlokat!

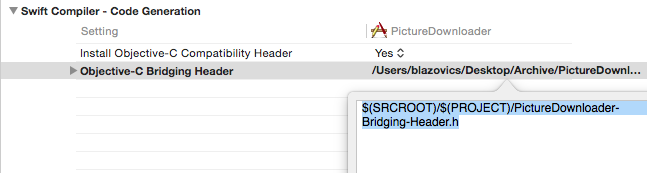


Mivel ezek Objective-C-ben írt állományok a rendszer automatikusan felajánlja, hogy készít hozzájuk bridging headert.

**Ha nem történnne meg**, akkor hozzunk létre egy **PictureDownloader-Bridging-Header.h** állományt majd menjünk át a projektbeállításokhoz, azon belül is Build settings-hez és állítsuk be a Brigding headert.

Ha a projekt gyökérmappájába tettük a fájlt, akkor ezt a sort kell belemásolni:

$(SRCROOT)/$(PROJECT)/PictureDownloader-Bridging-Header.h



Végül egészítsük ki a PictureDownloader-Bridging-Header.h-t az #import “MBProgressHUD.h” direktívával!

#import "MBProgressHUD.h"

Ezután egészítsük ki a ViewController-t, hogy letöltés közben kijelződjön a letöltés állapota.

Hozzunk létre egy MBProgressHUD típusú property-t, hogy később bárhonnan el tudjuk érni:

var contentUrl:NSURL?

var imageUrl:NSURL?

var imageView:UIImageView?

var HUD:MBProgressHUD?

override func viewDidLoad() {

Ezután inicializáljuk a viewDidLoad metódus végén.

self.navigationItem.leftBarButtonItem =

UIBarButtonItem(customView: button)

self.view.addSubview(displayButton)

HUD = MBProgressHUD(view: self.view)

self.view.addSubview(HUD!)

}

Amikor elindul a letöltés jelenítsük meg, a egyes adatcsomagok beérkezésekor frissítsük, végül, ha minden letöltődött, rejtsük el a HUD-ot.

Ehhez a downloadFile()-t kell kiegészíteni:

func downloadFile(){

let destination =

Alamofire.Request.suggestedDownloadDestination(

directory: .DocumentDirectory,

domain: .UserDomainMask)

Alamofire.download(

.GET, contentUrl!, destination: destination)

.progress {

bytesRead,

totalBytesRead,

totalBytesExpectedToRead in

self.HUD?.progress =

Float(totalBytesRead) /

Float(totalBytesExpectedToRead)

self.HUD?.labelText =

NSString(format: "%.0f%%",

Float(totalBytesRead) /

Float(totalBytesExpectedToRead)\*100) as String

}

.response { \_, \_, \_, error in

if let error = error {

print("error: \(error)")

} else {

print("success")

}

self.HUD?.hide(true)

}

HUD?.show(true)

}

## CocoaPods

Végül, hogy ne kelljen minden egyes nézetváltásokor letölteni a kisméretű képeket, adjunk hozzá a projekthez egy image cache komponenst, mely elvégzi ezt a feladatot.

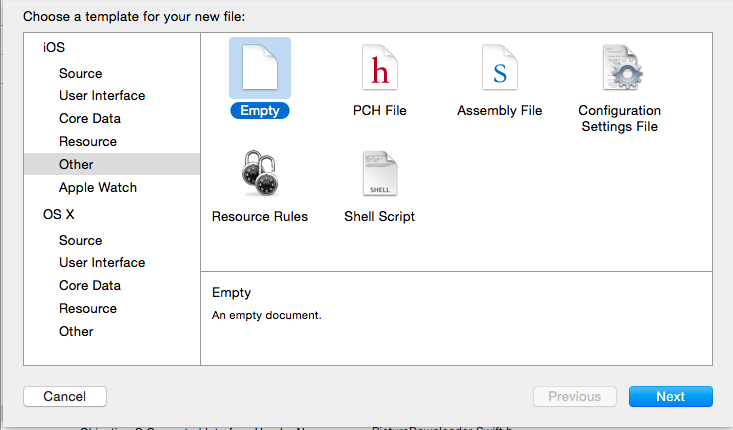
Ehhez a Kingfish nevű third-party modult fogjuk használni, amit a CocoaPods nevű keretrendszerrel fogunk hozzáadni a projekthez.

Először telepítsük/frissítsük a CocoaPods-ot. (**A laborgépeken nem lehetséges, mert adminisztrátori jogosultság kell hozzá!**)

Nyissuk meg a terminált és írjuk be a következő parancsot:

$ sudo gem install cocoapods

Majd adjunk hozzá egy üres fájt a projektünk gyökérkönyvtárába Podfile néven. Ez a legegyszerűbben úgy tudjuk megtenni, hogy a projekthez magához adunk hozzá egy üres fájlt.



A tartalma legyen a következő:

source 'https://github.com/CocoaPods/Specs.git'

platform :ios, '8.0'

use\_frameworks!

pod 'Kingfisher', '~> 1.8'

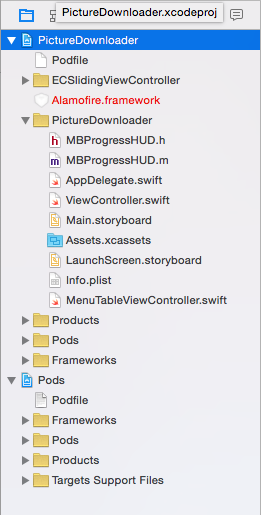
pod 'Alamofire', '~> 3.0'

Az Alamofire azért kell beleírni, hogy az újonan generált workspace is magába foglalja a modult. Mostantól viszont már akár le is töröhetjük a korábban letöltött állományokat.

Mentsük el, majd menjünk vissza terminálhoz navigáljunk el a **project gyökérkönyvtárába** és írjuk be a következő parancsot:

$ pod install

Ha sikerült, zárjuk be a projektet és a Finderben keressük meg a projektünket. Látható, hogy a PictureDownloader.xcodeproj projektfájl mellett megjelent egy PictureDownloader.xcworkspace fájl is. Utóbbit nyissuk meg.

Valami ilyesmit kell látnunk.

Nyissuk meg a ViewController.swift állományunkat és töröljük/kommentezzük ki a korábban írt viewWillAppear metódust.

Ezzel sajnos azt értük el, hogy már nem töltődik be automatikusan a járművek képei, amikor a nézetet megjelenítjük. Hogy a Kingfish-t tudjuk használni adjuk hozzá a következő sort a fájlhoz.

import UIKit

import Alamofire

import Kingfisher

Végül egészítsük ki a ViewController viewDidLoad metódusát, hogy le is töltödjön a kép. Ehhez adjunk hozzá a következő sort:

self.view.addSubview(displayButton)

HUD = MBProgressHUD(view: self.view)

self.view.addSubview(HUD!)

self.imageView!.kf\_setImageWithURL(imageUrl!)

}

## Önálló feladatok

1. Jelenítsük meg a letöltött képet valami kreatív módon!