iOS alapú szoftverfejlesztés - Labor 14

A laborsegédletet összeállította: Kántor Tibor - kantor.tibor@bmeautsoft.hu

A labor témája:

* Dátumkezelés
* Swift funkcionláis lehetőségei
* VisualEffectView
* Local Notifications
* TodayWidget
* Embedded framework

A labor célja az iOS extra funkcióiba való betekintés, továbbá a TodayWidget használatának a bemutatása.

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

https://gist.github.com/DjCantor/2088d03ef8442ee02092

# AutoLayout bemutatása

## Kiinduló project iNames

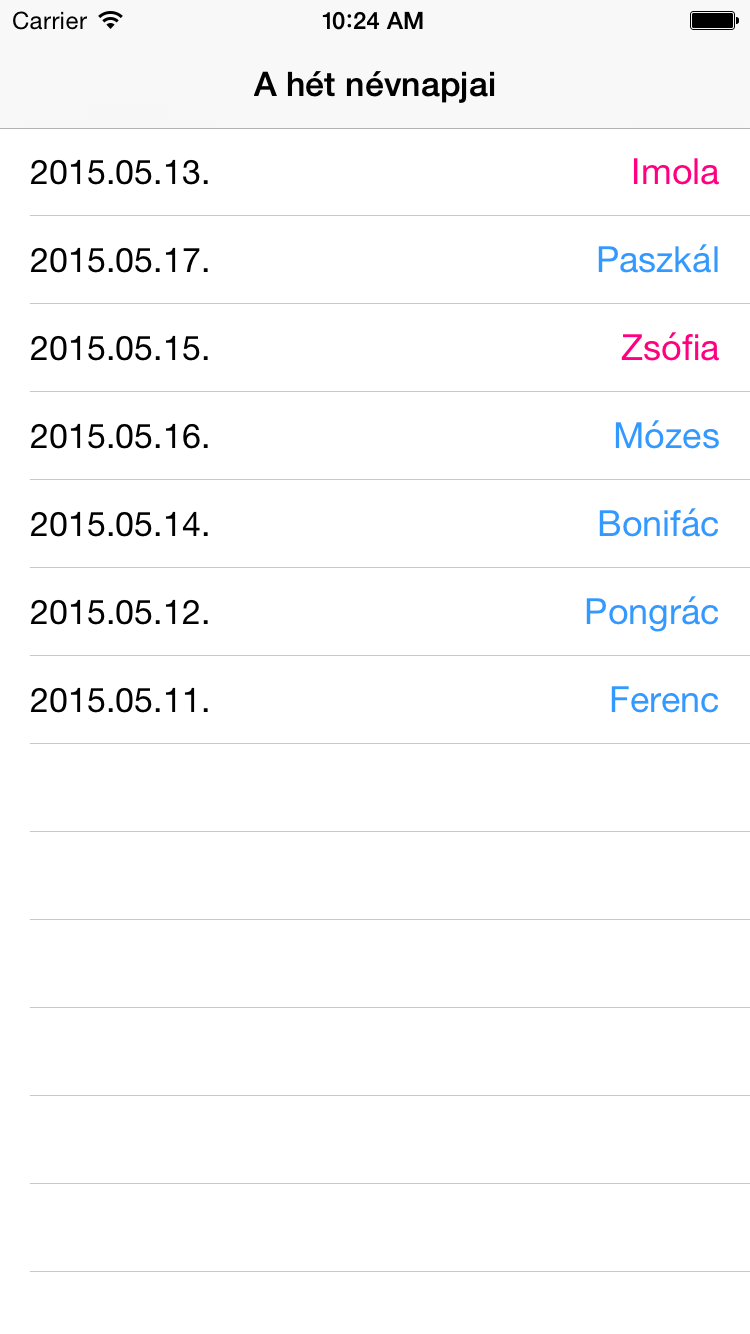
Töltsük le a kiinduló projektünket a következő URL-ről:

***https://www.dropbox.com/s/cdgace5imptx2zm/iNames.zip?dl=0***

A projekt a következő fájlokból áll:

* Names.plist: tartalmazza a neveket
* Name: egy nevet reprezentáló osztály
* NameManager: a nevek beolvasásáért és visszaaásááért felel
* NamesViewController: e heti névnapok

Indítsuk el az alkalmazást, amely a hét névnapjait jeleníti meg:



## Dátumok beolvasása és kiírása

Ha elindítjuk az alkalmazást, azt tapasztaljuk, hogy a dátumok előtt ott szerepel a 2015, noha ez egy teljesen felesleges információ, ráadásul sokkal, felhasználóbarátabb volna, ha pl. a hét napja lenne kiírva.

Ha most megnézzük a Name.swift fájlt láthatjuk, hogy jelenleg a dátumot egy stringbe olvassuk be.

Ahhoz, hogy ezt dátumként tudjuk kezelni, először is a típusát változtassuk meg NSDate opcionálisra:

var date : NSDate!

A beolvasó kódot pedig módosítsuk:

name = nameDictionary["name"] as! String

...

let dateString = nameDictionary["date"] as! String

let dateFormatter = NSDateFormatter()

dateFormatter.dateFormat = "yyyy.MM.dd.' 'HH:mm"

date = dateFormatter.dateFromString(dateString)

...

meaning = nameDictionary["meaning"] as! String

Ám ezzel elromlott a dátum megjelenítésért felelős kódunk!

Vegyünk fel a NamesViewControllerbe egy új dateFormatter tagváltozót és inicializáljuk a ViewDidLoad-ban:

let dateFormatter = NSDateFormatter()

dateFormatter.dateFormat = "dd, EEEE"

Végül a cellForRowAtIndexPath-t módosítsuk a piros sort:

cell.textLabel?.text = dateFormatter.stringFromDate(name.date)

## Sorrendezés, kiválasztás

Felhasználói szempontból elég zavaró, hogy nem sorrendben vannak a nevek. Rendezzük őket sorba! Ehhez vegyünk fel egy új függvényt, amely sorrendbe rakja őket a NameManager osztályban:

func dateSortedNames() ->[Name] {

return names.sort({ (name1: Name, name2: Name) -> Bool in return   
 name1.date!.compare(name2.date) == NSComparisonResult.OrderedAscending   
 })

}

A swift támogatja a funkcionális programozást! A funckionális programozás a deklaratív programnyelv egy speciális osztálya, amelyben a program alapvető építő kövei a függvények. Ezek a függvények nem eljárások, hanem két entitás közti kapcsolatot leíró függvények (mint egy matematikai egyenletben). A jelenkorban egyre nagyobb szerepet kap a funkcionális programozás, ugyanis az OOP nincs felkészítve az elosztott, konkurens működésére: általában szükséges valamilyen állapot (bonyolult) karbantartása, míg a funkcionális programozásban erre nincs szükség.

Próbáljuk ki!

names = NameManager.sharedInstance.dateSortedNames()

A helyzet az, hogy maga closure redukálható! A típusok kiírása nem tesz semmit hozzá:

return names.sorted({ (name1, name2) -> Bool in return   
 name1.date.compare(name2.date) == NSComparisonResult.OrderedAscending })

De a rendszer a visszatérési értéket is tudja és a zárójeleket is elhagyhatjuk!

return names.sorted({ name1, name2 in return   
 name1.date.compare(name2.date) == NSComparisonResult.OrderedAscending })

Sőt, valójában elég magát a rendező függvényt megadni, ekkor $ jelekkel hivatkozhatunk a paraméterekre!

return names.sorted({ $0.date.compare($1.date) ==   
 NSComparisonResult.OrderedAscending})

Jó lenne, ha visszaadná a NamesManager, hogy pontosan ma milyen névnap van és ezt jelezni tudnánk a felhasználónak. Ha feltesszük, hogy minden nap csak egy névnap van, akkor pl. szűrjük le a tömböt a mai napra és ennek a tömbnek az egyetlen elemét adjuk vissza!

func todaysName() -> Name {

let todayNames = names.filter {

let now = NSDate()

let calendar = NSCalendar(calendarIdentifier:   
 NSCalendarIdentifierGregorian)

let dateComponents = calendar?.components([.Year   
 .Month, .Day], fromDate: now)

dateComponents!.hour = 0

calendar?.timeZone = NSTimeZone(name: "GMT")!

let today0AM = calendar?.dateFromComponents(dateComponents!)

let currentDay = dateComponents?.day

dateComponents!.day = currentDay!+1

let today24AM = calendar?.dateFromComponents(dateComponents!)

if $0.date.compare(today24AM!) == NSComparisonResult.OrderedAscending   
 && today0AM!.compare($0.date) == NSComparisonResult.OrderedAscending {

return true

}

return false

}

return todayNames[0]

}

https://gist.github.com/DjCantor/2088d03ef8442ee02092#file-inames-todaysname

Ennek megfelelően egészítsük ki a NamesViewController-t! Ehhez vegyük fel tagváltozónak a mai nevet todaysName néven, amit a viewDidLoad-ban inicializáljunk az iménti függvénnyel. A cella visszaadó függvényt meg módosítsuk, hogy az adott sorban pl. legyen más a felirat:

if todaysName == name {

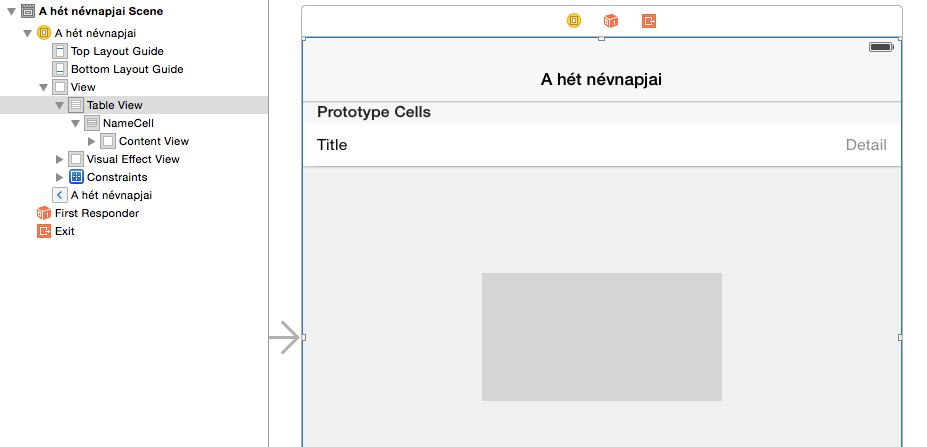
cell.detailTextLabel?.textColor = UIColor.greenColor()

cell.textLabel?.text = "MA!"

}

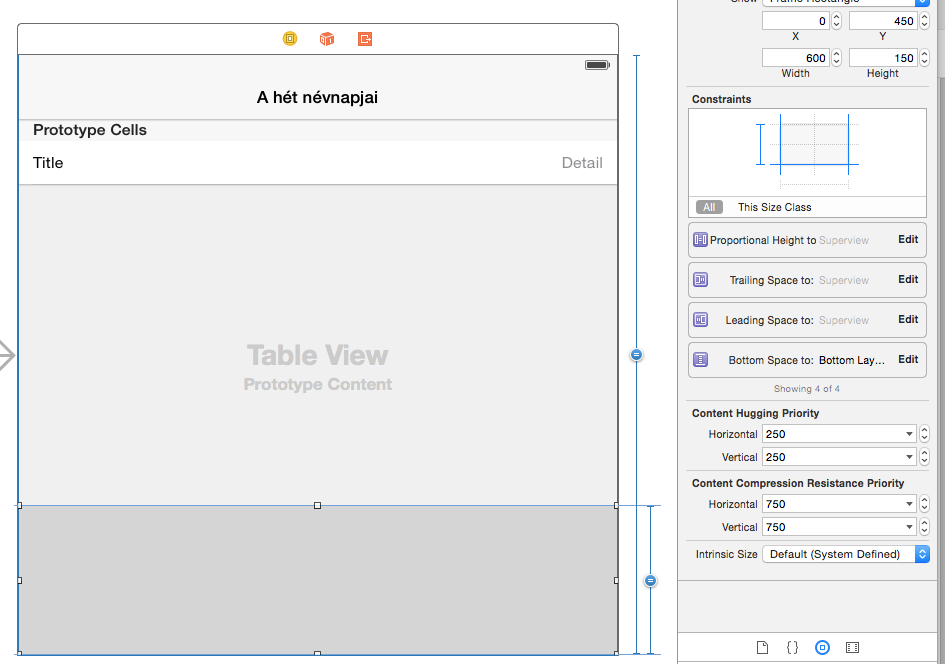
## Vizuális effektusok

Tegyünk be a Storyboard-ban egy VisualEffectView With Blur And Vibrancy-t a táblázat fölé a Document Outline-ban ügyelve a Z orderre, és hogy ne, hogy a táblázatba kerüljön!



Nézzük meg a berakott View hierarchiát! A legfelső nézet felel a „blurözésért”, az alsó pedig a rárakott UIControllok „vibrálásáért” (kiemeléséért).

Módosítsuk a kényszereit a szülő aljához és oldalához, a magassága pedig mindig legyen a szülő nézet magasságának a negyede!



Próbáljuk ki az alkalmazást, célszerű kisebb kijelzőn!

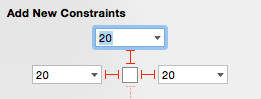
Hogy jobban lássuk a működést érdemes megnövelni a cella méretet:

func tableView(tableView: UITableView,   
 heightForRowAtIndexPath indexPath: NSIndexPath) -> CGFloat {

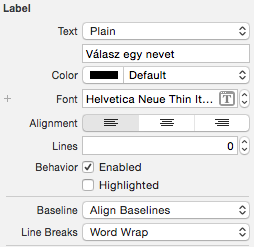
return 88

}

Tegyünk be egy labelt az effekt nézet közepére a következő következő kényszerekkel:



A labelen alkalmazzuk a következő beállításokat (Helvetica Neue Thin Itallic 19, Lines: 0, Line Breaks 0, felirate: Válassz egy nevet!



Vegyünk fel egy descriptionLabel nevű outletet, majd valósítsuk meg a didSelectRowAtIndexPath függvényt:

func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {

let name = names[indexPath.row]

self.descriptionLabel.text = "Jelentése:   
 \(name.meaning)\nEredete:\(name.origin)"

}

Koncentráljunk a labelre és próbáljuk ki az alkalmazást. Érdemes úgy is kipróbálni, hogy a belső Visual Effect View-n kikapcsoljuk a Vibrancy-t!

## Lokális értesítés

Tegyünk be egy gombot a label alá középre, aminek a felirata legyen „Értesíts!” és adjuk, meg, hogy mindig 20 egységre legyen a nézet aljától!

Ahhoz, hogy értesíteni tudjuk a felhasználót, engedélyt kell kérnünk. Egészítsük ki az ApplicationDidFinishWithOptions:

application.registerUserNotificationSettings(UIUserNotificationSettings(forTypes: [.Sound, .Alert, .Badge], categories: nil))

Ezek után írjuk meg a gomb esemény kezelőjét, hogy 10 másodperc múlva értesítsen!

@IBAction func notifyButtonTouchUpInside(sender: AnyObject) {

var localNotification: UILocalNotification = UILocalNotification()

localNotification.alertAction = "Reminder for name"

localNotification.alertBody = "Woww it works!!"

localNotification.fireDate = NSDate(timeIntervalSinceNow: 10)

UIApplication.sharedApplication().scheduleLocalNotification(  
 localNotification)

}

Próbáljuk ki az alkalmazást! Ne felejtsük el háttérbe küldeni az appot!

Ha előtérben van az alkalmazás, és értesítést kaptunk akkor ezt az AppDelegate-ben tudjuk lekezelni:

func application(application: UIApplication, didReceiveLocalNotification notification: UILocalNotification) {

let alertController = UIAlertController(title: "Reminder for name!",   
 message: "", preferredStyle: .Alert)

let OKAction = UIAlertAction(title: "Ok", style: .Default) { action in }

alertController.addAction(OKAction)

let vc = UIApplication.sharedApplication().keyWindow?.rootViewController

vc!.presentViewController(alertController, animated: true,   
 completion: nil)

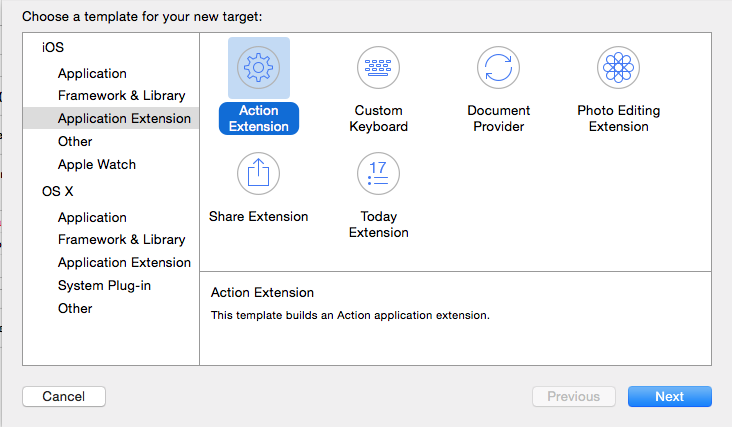
}

https://gist.github.com/DjCantor/2088d03ef8442ee02092#file-inames-didreceivelocalnotification

Próbáljuk ki ezt is!

## Today Widget

Adjunk hozzá egy új Today Extension targeted a projekthez (New Target / Today Extension) a neve legyen NameOfDay!

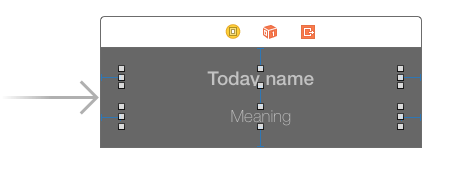


Ekkor a rendszer megkérdezi, hogy aktiválja-e a TodayExtension-höz tartozó sémát, engedélyezzük!

Próbáljuk ki az alkalmazást, kattintsunk a buld & run-ra, ilyenkor elindul a megfelelő szimulátor a NotificationCenter-rel!

Menjünk át Widget storyboardjába, töröljük a Hello World-öt és a következő képpen módosítsuk a layoutot:

Növeljük meg a méretet (99-re) majd tegyünk be két labelt: mindkettő legyen középre rendezve és a margók legyen 20 egységnyiek! A felső betűtípusa legyen Helvetica Neue Medium 19, az alsóé pedig Helvetica Neue Thin 17! A felso compression resistance priority-ja legyen 749, hogy eltűntessük a warningot (különben a rendszer nem tudná, hogy mit hogyan nyomjon össze, ha túl nagy a szöveg) A színük meg legyen világos szürke!



Ezekhez vegyünk fel outleteket is nameLabel és meaningLabel néven!

Kérdés, hogyan töltjük fel tartalommal?

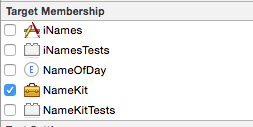
## Embeded Framework

Adjunk hozzá egy új targeted, ám ezúttal Framework-öt! Legyen a neve NameKit!

Húzzuk át a modell fájljainkat a Kit-be:

Macintosh HD:Users:kantortibor:Desktop:Screen Shot 2015-05-14 at 00.06.48.png

Állítsuk be, hogy ennek a targetnek a részei legyenek a file Inspectorban!



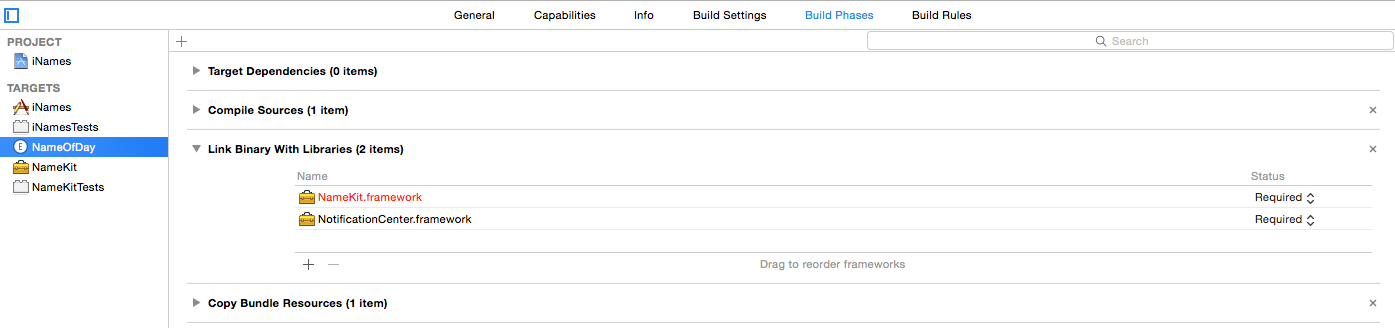
Importáljuk be a NamesViewController-be a NameKit-et és próbáljuk ki az alkalmazást!

Nem fordul, ennek az oka, hogy az egyes elemek láthatósága alapból internal (ami azt jelenti, hogy csak az adott targetből láthatók!) Ennek a feloldásához írjunk a változók és osztály neve elé, hogy public!

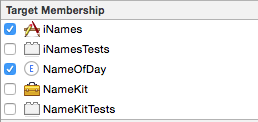
Próbáljuk ki az alkalmazást!

Ahhoz, hogy a widget-ből is elérhető legyen a következőket kell tennünk:

A WidgetTarget-nél állítsuk be, hogy linkelje a NameKit-et is!



A Names.plist fájlt pedig adjuk hozzá a NameOfDay targethez is:



Importáljuk be a TodayViewController-be a NameKit-et

Majd a ViewDidLoad-ot egészítsük ki:

override func viewDidLoad() {

super.viewDidLoad()

// Do any additional setup after loading the view from its nib.

let todayName = NameManager.sharedInstance.todaysName()

self.nameLabel.text = todayName.name

self.meaningLabel.text = "Jelentése: \(todayName.meaning)"

}

Próbáljuk ki az alkalmazást!

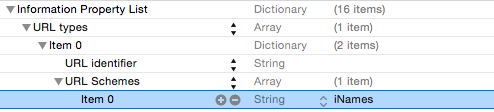
Ahhoz, hogy a bal oldali margót eltűntessük valósítsuk meg a margókat kezelő delegate függvényt:

func widgetMarginInsetsForProposedMarginInsets(  
 defaultMarginInsets: UIEdgeInsets) -> UIEdgeInsets {

return UIEdgeInsetsZero

}

Ahhoz, hogy kattintásra megnyíljon a host alkalmazás vegyünk fel neki egy új URL sémát a plist fájlban (iNames / Info.plist):



Majd tegyünk fel egy widget méretű gombot a widgetre, a típusa legyen custom, szövege pedig ne legyen!

Ennek az eseményvezérlője pedig:

@IBAction func openHostAppButtonTouchUpInside(sender: AnyObject) {

var url: NSURL = NSURL(string: "iNames://")!

self.extensionContext?.openURL(url, completionHandler: nil)

}

https://gist.github.com/DjCantor/2088d03ef8442ee02092#file-inames-openhost

# Önálló feladatok pluszpontért

## Értesítés időpontja

Valósítsuk meg, hogy az értesítés aznap reggel 8-kor jöjjön, amikor az adott névnap van, ehhez használjuk fel a tömb rendezéséhez szükséges kódból a dátum módosítást! Println-nel írjuk ki a kész notification-t és ezt mutassuk meg a laborvezetőnek!

## Táblázat görgetés

Valósítsuk meg, hogy a táblázat alját is látni lehessen: ehhez a táblázat contentInset propertyjétt kell beállítani, még pedig az alsó értéket (tipp: a visual magassát mi is ki számolhatjuk a kódban (a képernyő magassága pedig lekérdezhető)!.