iOS alapú szoftverfejlesztés - Labor 10

A laborsegédletet összeállította: Kelényi Imre - [imre.kelenyi@aut.bme.hu](mailto:imre.kelenyi@aut.bme.hu), és Kántor Tibor

A labor témája:

* Hálózati kommunikációs és web API-k használata
* JSON feldolgozás

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

https://gist.github.com/imrekel/ac64ad22ad5ed4a1db26

# Messenger

Töltsük le a kiindulási állapotban lévő projektet:

**https://dl.dropbox.com/u/152439/aut-ios/Messenger-start.zip**

Próbáljuk ki az alkalmazást és nézzük át a forráskódját! Az alkalmazás tartalmaz egy Table View Controller-t (MessagesViewController), illetve egy új üzenetek írására szolgáló ComposeMessageViewController-t.

## Üzenetek letöltése

MessagesViewController-be vegyünk fel egy új property-t, mely az NSURLSession példányt tárolja:

var urlSession: NSURLSession!

Inicializáljuk a sessiont viewDidLoad()-ban:

override func viewDidLoad() {

super.viewDidLoad()

let sessionConfig =   
 NSURLSessionConfiguration.defaultSessionConfiguration()

urlSession = NSURLSession(configuration: sessionConfig, delegate: nil,   
 delegateQueue: NSOperationQueue.mainQueue())

}

Valósítsuk meg a Refresh gomb megnyomásakor meghívódó metódust, mely elindítja az üzenetek letöltését (az üres metódus már ott van a kódban és be is van kötve a gomb megfelelő eseményéhez).

@IBAction func refreshButtonTap(sender: AnyObject) {

let url = NSURL(string:   
 "http://atleast.aut.bme.hu/ait-ios/messenger/messages")

let dataTask = urlSession.dataTaskWithURL(url!, completionHandler: {   
 data, response, error in

if let data = data {

let responseString = NSString(data: data, encoding:

NSUTF8StringEncoding)

print(responseString)

}

})

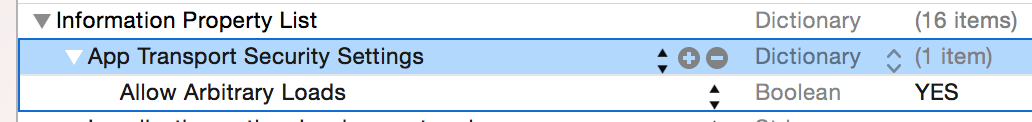
dataTask.resume()

}

Teszteljük az alkalmazást és ellenőrizzük, hogy a konzolon megjelenik a letöltött JSON formátumú válasz!

Az első indításkor egy warningot fogunk kapni, mivel egy sima HTTP kapcsolatot próbálunk felépíteni (nem pedig egy megfelelően biztonságos HTTPS-t) és nincs feloldva az iOS 9-en bevezetett App Transport Security.

Nyissuk meg az app Info.plist fájlját és vegyük fel a megfelelő beállításokat az ATS kikapcsolásához:



## JSON feldolgozás

A szervertől kapott válasz JSON formátumú. Egy külső tömbben JSON objektumok írják le a megjelenítendő üzeneteket. A szerver válaszát egy böngészőben is megvizsgálhatjuk az URL megnyitásával:

**http://atleast.aut.bme.hu/ait-ios/messenger/messages**

[

{"from\_user":"The Nameless One",

"to\_user":"Someone",

"topic":"Keep calm and...",

"content":"",

"imageurl":"http://server.com/noimage.png",

"latitude":47.562598,

"longitude":19.054155

}

]

Az üzenetek tárolásához vegyünk fel egy új property-t, melyet egy üres tömbbel inicializáljunk:

var messages = [AnyObject]()

A data task befejeztekor meghívódó blokkban dolgozzuk fel a kapott JSON-t és rendeljük az eredményt a messages tagváltozóhoz:

do {

guard let messages = try NSJSONSerialization.JSONObjectWithData(data!,

options: NSJSONReadingOptions(rawValue: 0)) as? [AnyObject] else {

return

}

self.messages = messages

self.tableView.reloadData()

} catch {

print("Error parsing JSON")

}

Valósítsuk meg a table view adatforrás metódusait, hogy megjelenítsük az üzeneteket:

override func tableView(tableView: UITableView, numberOfRowsInSection   
 section: Int) -> Int {

return messages.count

}

Az üzenetek megjelenítéséhez egy egyedi cella osztályt használunk (MessageCell):

override func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCellWithIdentifier("MessageCell",   
 forIndexPath: indexPath) as! MessageCell

let message = messages[indexPath.row] as! [NSObject: AnyObject]

let fromUser = message["from\_user"] as! String

let toUser = message["to\_user"] as! String

let topic = message["topic"] as! String

cell.recipientLabel.text = fromUser + " - " + toUser

cell.subjectLabel.text = topic

return cell

}

https://gist.github.com/imrekel/ac64ad22ad5ed4a1db26#file-cellforrowatindexpath-swift

## Üzenetek feltöltése

Új üzenet küldéséhez a url-re kell egy HTTP POST kérést küldenünk, a következő formátumú tartalommal:

{

"from\_user" : "",

"to\_user" : "",

"topic" : "",

"content" : "",

"image" : "Base64 kódolású JPEG kép"

}

Az üzenet összeállítását a composeViewControllerDidSend() metódusban végezhetjük, mely be van kötve az új üzenet írásához készített view controllerhez.

Első lépésként állítsuk össze az adathierarchiát, majd alakítsuk JSON-ná (az esetleg megadott kép feltöltését későbbre hagyjuk). A “YOUR NAME” helyett mindenki válasszon egy egyedi nevet:

var message = [NSObject: AnyObject]()

message["from\_user"] = "YOUR NAME"

message["to\_user"] = viewController.toUserTextField.text

message["topic"] = viewController.topicTextField.text

let jsonData: NSData!

do {

jsonData = try NSJSONSerialization.dataWithJSONObject(message,

options: NSJSONWritingOptions(rawValue: 0))

} catch {

return

}

A POST kérés küldéséhez már egy NSMutableURLRequestre less szükségünk:

let url = NSURL(string:   
 "http://atleast.aut.bme.hu/ait-ios/messenger/add-message")

let request = NSMutableURLRequest(URL: url!)

request.HTTPMethod = "POST"

request.setValue("application/json", forHTTPHeaderField: "Content-Type")

Indítsunk egy upload taskot, mely befejeztekor egy alertet feldobva nyugtázzuk a folyamatot:

let postTask = urlSession.uploadTaskWithRequest(request, fromData:   
 jsonData, completionHandler: {data, response, error in

do {

guard let response = try NSJSONSerialization.JSONObjectWithData(data!,   
 options: NSJSONReadingOptions(rawValue: 0)) as? [NSObject: AnyObject]   
 else {

return

}

let result = response["result"] as! String

let alert = UIAlertController(title: "Server response", message:   
 result, preferredStyle: .Alert)

let okAction = UIAlertAction(title: "Ok", style: .Default,   
 handler: nil)

alert.addAction(okAction)

self.presentViewController(alert, animated: true, completion: nil)

} catch {}

})

postTask.resume()

https://gist.github.com/imrekel/ac64ad22ad5ed4a1db26#file-upload-message-swift

Ha újra letöltjük az üzeneteket, meg kell jelennie az új küldeménynek.

## Képek feltöltése

A szervernek elküldendő JSON üzenetbe illesszük be a kiválasztott képet. Ehhez először lekicsinyítjük, majd a JPEG reprezentációját base64 kódolással alakítjuk sztringgé:

if let image = viewController.imageView.image {

let scaledImage = image.scaledImageToSize(CGSize(width: 40, height: 40))

if let jpegImageData = UIImageJPEGRepresentation(scaledImage, 0.7) {

message["image"] = jpegImageData.base64EncodedStringWithOptions(  
 NSDataBase64EncodingOptions(rawValue: 0))

}

}

## Képek letöltése

A szerveren minden feltöltött kép eltárolódik, majd az üzenetek lekérdezésekor az „imageurl” kulcshoz tartozó érték alapján tudjuk letölteni őket. Hogy ne töltsünk le feleslegesen egy képet többször, hozzunk létre egy kis „cache”-t, mely URL – kép párokat tartalmaz. Vegyünk fel egy új property-t MessagesViewController-hez:

// Holds images associated with their URL

var imageCache = [NSURL: UIImage]()

Definiáljunk egy új metódust, mely az URL-je alapján beállít egy képet a cacheből egy cellához, vagy letölti, ha még nincs meg és azután állítja be:

// MARK: - New methods

func downloadImageWithUrl(url: NSURL, forTableViewCell cell:   
 MessageCell) {

if let cachedImage = imageCache[url] {

cell.messageImageView.image = cachedImage

}

else {

cell.messageImageView.image = nil

let imageDownloadTask = urlSession.dataTaskWithURL(url,   
 completionHandler: {[weak cell] data, response, error in

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

self.imageCache[url] = image

cell?.messageImageView.image = image

}

})

imageDownloadTask.resume()

}

}

https://gist.github.com/imrekel/ac64ad22ad5ed4a1db26#file-downloadimagewithurl-swift

A cellák létrehozásakor (cellForRowAtIndexPath()) indítsuk el a cellához tartozó kép letöltését:

if let imageURLString = message["imageurl"] as? String {

downloadImageWithUrl(NSURL(string: imageURLString)!,   
 forTableViewCell: cell)

}

## Network Activity Indicator

Jelenítsük meg ill. rejtsük el a hálózati aktivitást jelző „pörgettyűt”, üzenet küldésekor és üzenetek letöltésekor, illetve mikor a műveletek véget érnek:

UIApplication.sharedApplication().networkActivityIndicatorVisible = true

UIApplication.sharedApplication().networkActivityIndicatorVisible = false

# Önálló feladat: Valutaváltó

Készítsünk egy egyszerű valutaváltó alkalmazást, a <http://fixer.io> API-t használva:

* Hozzunk létre egy új Single View alkalmazás
* Készítsünk egy egyszerű felhasználói felületet (szükség lesz text fieldekre a valutanemek és az átváltandó összeg bekérése, labelre az eredmény kiírásához, valamint egy gombra a folyamat indításához)
* Az átváltás gomb megnyomásakor indítsunk egy HTTP GET kérést (egy data taskot), mely letölti az aktuális árfolyamot. Az URL formátuma a következő:  
   http://api.fixer.io/latest?base=**USD**&symbols=**HUF**
* Dolgozzuk fel a JSON választ és jelenítsük meg a váltás eredményét!
  + A számértékek NSNumber-őkre képződnek le, amiket lehet castolni pl. Double-ra (as Double)
  + A válaszban a váltási valutanem lesz az egyik kulcs érték

{

"base":"USD",

"date":"2015-04-07",

"rates": **{**

**"HUF":275.73**

**}**

}