Advanced Mobile Application Development
Week 7: Local Notifications

todo
File | New Project
Single View App
iPhone

Setup
Delete the ViewController.swift file.
Go into the Storyboard and delete the view controller.
Add a table view controller and embed it in a navigation controller.
Make the navigation controller the Initial View Controller.
Give the table view controller the title "To Do List".
Add a bar button on the right and change System Item to Add.
For the table view cell make the style Subtitle and give it a reuse identifier "Cell".
Add a View Controller to the right of the Table View Controller and embed it in a navigation controller.
Give it the title "To Do Item".
Add a bar button item on the right side of the navigation bar and change it to Save.
Add a bar button item on the left side of the navigation bar and change it to Cancel.
Add a label and textfield for the user to add the item.
Add a connection for the textfield called itemtextfield.
Add a label that says "Reminder" and a date picker under it.
Add a connection for the date picker called reminderDatePicker.
Use auto layout to add needed constraints for this view.

Add two Cocoa touch classes to control these.
Call the first ToDoTableViewController and subclass TableViewController.
Call the second ItemViewController and subclass ViewController.

Back in the storyboard change the two views to use these classes.
Create a show segue from the Add bar button (use the document hierarchy) to the Navigation Controller for the Item View Controller.

In order to navigate back create an unwind method in ToDoTableViewController.swift

```
@IBAction func unwindSegue(segue:UIStoryboardSegue){
}
```

In the storyboard connect the Cancel and Save button to the Exit icon and chose the unwindSegue method. Name the segues "cancel" and "save".

You should be able to run it and navigate back and forth.

Keyboard
When you enter a new item you'll notice the keyboard can't be dismissed. This is especially a problem in landscape mode where the keyboard covers the picker.
To get the return key to dismiss the keyboard go into ItemViewController.swift and adopt the UITextFieldDelegate protocol and set the delegate to self in viewDidLoad()

```
        itemtextfield.delegate=self
```

Then implement the method that's called when the return key is pressed.

```
    func textFieldShouldReturn(textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }
```

Data

Create a swift class called TodoItem to store our to do items. (new swift file)

```
class TodoItem {
    var name : String
    var reminderDate : NSDate
    var id : String

    init(newName: String, newReminderDate: NSDate){
        self.name = newName
        self.reminderDate = newReminderDate
        id = newId
    }

    func overDue()->Bool {
        // reminder date is earlier than current date
        return (NSDate().compare(self.reminderDate) ==
NSComparisonResult.OrderedDescending)
    }

}
```

Our class needs at least one initializer method.

We'll use the overdue method to determine if an item is past its due date.

In ItemViewController create a variable to hold a new item

```
    var addedItem : TodoItem?
```

Now we can implement prepareForSegue

```
    override func prepareForSegue(segue: UIStoryboardSegue, sender:
AnyObject?) {
        if segue.identifier == "save" {
            if ((itemtextfield.text?.isEmpty) == false){  //save new item
                // initialize added item
                addedItem = TodoItem(newName: itemtextfield.text!,
newReminderDate: reminderDatePicker.date, newId: NSUUID().UUIDString)
            }
        }
    }
```

In ToDoTableViewController create an array to hold all our items and implement unwindSegue

```
var items = [TodoItem]()

    @IBAction func unwindSegue(segue:UIStoryboardSegue){
        if segue.identifier == "save" {
            //save new item
            let source = segue.sourceViewController as! ItemViewController
```

```
            if let newItem = source.addedItem{
                items.append(newItem)
                tableView.reloadData()
            }
        }
    }
```

Implement the table view data source methods

```
   override func numberOfSectionsInTableView(tableView: UITableView) -> Int
{
        // #warning Incomplete implementation, return the number of sections
        return 1
    }

    override func tableView(tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        // #warning Incomplete implementation, return the number of rows
        return items.count
    }

    override func tableView(tableView: UITableView, cellForRowAtIndexPath
indexPath: NSIndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("Cell",
forIndexPath: indexPath)

        // Configure the cell...
        cell.textLabel?.text=items[indexPath.row].name
        if items[indexPath.row].overDue() {
            cell.detailTextLabel?.textColor = UIColor.redColor()
        } else {
            // we need to reset this because a cell with red subtitle may be
returned by dequeueReusableCellWithIdentifier
            cell.detailTextLabel?.textColor = UIColor.blackColor()
        }
        let dateformatter = NSDateFormatter()
        dateformatter.dateFormat="MMM dd yyyy hh:mm a"
        let date = items[indexPath.row].reminderDate
        cell.detailTextLabel!.text=dateformatter.stringFromDate(date)

        return cell
    }
```

We use the NSDateFormatter class to format the date from the date picker and turn it into a string for our cell detail label.

Register Notification Settings

As of iOS 8, we need to request permission from the user to allow notifications.
AppDelegate.swift

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

```
application.registerUserNotificationSettings(UIUserNotificationSettings(forT
ypes: [.Alert, .Badge, .Sound], categories: nil))
        return true
    }
```

And let's reload the data in the table every time the view appears so overdue items are red.

```
    override func viewDidAppear(animated: Bool) {
        tableView.reloadData()
    }
```

Now if you run it you will be prompted to allow notifications.

Schedule Notifications

Define a method that will schedule notifications.

```
    func addNotification(item: TodoItem){
        //check if notifications are enabled
        let settings =
UIApplication.sharedApplication().currentUserNotificationSettings()
        if settings?.types.rawValue == 0 {
            print("notifications not enabled")
        } else { //schedule notification
            let notification = UILocalNotification()
            notification.fireDate = item.reminderDate
            notification.alertBody = item.name
            notification.alertAction = "open"
            notification.soundName = UILocalNotificationDefaultSoundName
            notification.userInfo = ["title": item.name, "UUID": item.id]
UIApplication.sharedApplication().scheduleLocalNotification(notification)
        }
    }
```

This will result in a notification if the app is NOT in the foreground, so either lock screen, home screen, or in another app. In the home screen or another app the notification will be a banner at the top of the screen and automatically go away, or an alert (based on notification setting in Settings). In the lock screen it will be a banner on the lock screen. Tapping/swiping any of these take you into the app.

We will call this from unwindSegue after we added the item to our table

```
addNotification(newItem)
```

Cmd+L to lock the iOS Simulator screen.

Go into Settings and scroll down to see your app. This is where the user can access and change their notification settings.

If you want to test allowing or denying permission, just reset the simulator and run the app again to get a clean slate. Simulator | Reset Content and Settings

It would also be nice to remind the user when they go to add a new item that a notification won't be scheduled because the app doesn't have permissions.

In ItemViewController add a method to do this.

```
    @IBAction func checkNotify() {
        let settings =
UIApplication.sharedApplication().currentUserNotificationSettings()
        if settings?.types.rawValue == 0 {
            let alert = UIAlertController(title: "Can't schedule
notifications", message: "Please go to Settings to enable notifications",
preferredStyle: .Alert)
            let action = UIAlertAction(title: "OK", style: .Default,
handler: nil)
            alert.addAction(action)
            presentViewController(alert, animated: true, completion: nil)
        }
    }
```

Call it from viewDidLoad

```
checkNotify()
```

I couldn't call this method while either view was segueing, the view must be completely loaded before you can present a view controller.

I also could have disabled the Save button and not saved the new item to the table. But I decided that some users might want the list of items in the table even with notifications turned off.

Schedule Notification for Foreground

The app delegate method application(_, didReceiveLocalNotification) is called to process notifications when the app is running. The system does not display any notifications when the app is in the foreground so you must create it yourself.

```
    func application(application: UIApplication, didReceiveLocalNotification
notification: UILocalNotification) {
           let appname =
NSBundle.mainBundle().infoDictionary!["CFBundleName"] as! String
            let alert = UIAlertController(title: appname, message:
notification.alertBody, preferredStyle: .Alert)
            let action = UIAlertAction(title: "OK", style: .Default,
handler: nil)
            alert.addAction(action)
            window?.rootViewController?.presentViewController(alert,
animated: true, completion: nil)
}
```

You'll notice that the table doesn't get refreshed after a notification in the foreground or when becoming active (from the background to foreground).

We're going to use the NSNotificationCenter to register a notification for these cases so we can reload the table.

Note that even though this has the word 'notification', it's not related to UINotification

Add this observer in viewDidLoad()

```
        NSNotificationCenter.defaultCenter().addObserver(self, selector:
"refreshList", name: "ListShouldRefresh", object: nil)
```

Then create the refreshList() method

```
    func refreshList(){
        tableView.reloadData()
    }
```

In the AppDelegate we need to post this notification in didReceiveLocalNotification and applicationDidBecomeActive

```
NSNotificationCenter.defaultCenter().postNotificationName("ListShouldRefresh
", object: self)
```

Removing items

Now let's remove items we don't want in our list anymore as well as their notification.

```
    func removeNotification(item: TodoItem){
        for notification in
UIApplication.sharedApplication().scheduledLocalNotifications! as
[UILocalNotification]{
            if notification.userInfo!["UUID"] as! String == item.id {
                //cancel the notification

UIApplication.sharedApplication().cancelLocalNotification(notification)
            }
        }
    }
```

Uncomment the stub method tableView(_, canEditRowAtIndexPath) and make sure it returns true. Uncomment an implement deletion from our array and table.

```
    override func tableView(tableView: UITableView, commitEditingStyle
editingStyle: UITableViewCellEditingStyle, forRowAtIndexPath indexPath:
NSIndexPath) {
        if editingStyle == .Delete {
            // remove the notification
            removeNotification(items[indexPath.row])
            // Delete the row from the data source
            items.removeAtIndex(indexPath.row)
            // delete item from table view
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation:
.Fade)
        }
}
```

Badge icon

Now lets get the badge icon up to date. Unfortunately this is a bit trickier than you might think. If you set the applicationIconBadgeNumber property at the time you create your notification you are setting based on that point in time and you don't know how many notifications there really will be when it fires. There's also no way to have the app increment the badge when the notification fires because it might be in the background. So I created a method that computes how many items are overdue and then call it whenever I add or remove a notification.

```
    func setBadgeNumber(){
        var badgeNumber = 0
        // count all overDue items
        for item in items {
            if item.overDue() {
                badgeNumber++
            }
        }
        // assign to the app icon badge number
        UIApplication.sharedApplication().applicationIconBadgeNumber =
badgeNumber
    }
```

Call `self.setBadgeNumber()` after deleting a row from the table(after items array is updated), after adding a notification, and at the end of refreshList()