Lecture 9 Demo Code: Smashtag Table View

Objective

Included below is the source code for the demo in lecture. It is provided under the same Creative Commons licensing as the rest of CS193p's course materials. The code which has not changed since the previous lecture is included but is grayed out. The Twitter framework code is available via a link in the Assignment 4 writeup document. And here is the project (not including the Twitter framework or the workspace you'll need to contain them both).

```
//
//  TweetTableViewController.swift
//  Smashtag
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit
import Twitter

// this entire project will not work
// unless you make a Workspace that includes
// both this application
// and the Twitter project
// and you drag the Product of the Twitter framework build
// into the Embedded Binaries section
// of the Project Settings of this application

class TweetTableViewController: UITableViewController, UITextFieldDelegate
{
    // MARK: Model

    // part of our Model
    // each sub-Array of Tweets is another "pull" from Twitter
    // and corresponds to a section in our table
    private var tweets = [Array<Twitter.Tweet>]()

    // public part of our Model
    // when this is set
    // we'll reset our tweets Array
    // to reflect the result of fetching Tweets that match
    var searchText: String? {
        didSet {
            searchTextField?.text = searchText
            searchTextField?.resignFirstResponder()
            lastTwitterRequest = nil                // REFRESHING
            tweets.removeAll()
            tableView.reloadData()
            searchForTweets()
            title = searchText
        }
    }
```

```
// MARK: Updating the Table

// just creates a Twitter.Request
// that finds tweets that match our searchText
private func twitterRequest() -> Twitter.Request? {
    if let query = searchText, !query.isEmpty {
        return Twitter.Request(search: query, count: 100)
    }
    return nil
}

// we track this so that
// a) we ignore tweets that come back from other than our last request
// b) when we want to refresh, we only get tweets newer than our last request
private var lastTwitterRequest: Twitter.Request?

// takes the searchText part of our Model
// and fires off a fetch for matching Tweets
// when they come back (if they're still relevant)
// we update our tweets array
// and then let the table view know that we added a section
// (it will then call our UITableViewDataSource to get what it needs)
private func searchForTweets() {
    // "lastTwitterRequest?.newer ??" was added after lecture for REFRESHING
    if let request = lastTwitterRequest?.newer ?? twitterRequest() {
        lastTwitterRequest = request
        request.fetchTweets { [weak self] newTweets in      // this is off the main queue
            DispatchQueue.main.async {                      // so dispatch back to main queue
                if request == self?.lastTwitterRequest {
                    self?.tweets.insert(newTweets, at:0)
                    self?.tableView.insertSections([0], with: .fade)
                }
                self?.refreshControl?.endRefreshing()   // REFRESHING
            }
        }
    } else {
        self.refreshControl?.endRefreshing()            // REFRESHING
    }
}

// Added after lecture for REFRESHING
@IBAction func refresh(_ sender: UIRefreshControl) {
    searchForTweets()
}

// MARK: View Controller Lifecycle

override func viewDidLoad() {
    super.viewDidLoad()
    // we use the row height in the storyboard as an "estimate"
    tableView.estimatedRowHeight = tableView.rowHeight
    // but use whatever autolayout says the height should be as the actual row height
    tableView.rowHeight = UITableViewAutomaticDimension
    // the row height could alternatively be set
    // using the UITableViewDelegate method heightForRowAt
}
```

```
    // MARK: Search Text Field

    // set ourself to be the UITextFieldDelegate
    // so that we can get textFieldShouldReturn sent to us
    @IBOutlet weak var searchTextField: UITextField! {
        didSet {
            searchTextField.delegate = self
        }
    }

    // when the return (i.e. Search) button is pressed in the keyboard
    // we go off to search for the text in the searchTextField
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        if textField == searchTextField {
            searchText = searchTextField.text
        }
        return true
    }

    // MARK: - UITableViewDataSource

    override func numberOfSections(in tableView: UITableView) -> Int {
        return tweets.count
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return tweets[section].count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "Tweet", for: indexPath)

        // get the tweet that is associated with this row
        // that the table view is asking us to provide a UITableViewCell for
        let tweet: Tweet = tweets[indexPath.section][indexPath.row]

        // Configure the cell...
        // the textLabel and detailTextLabel are for non-Custom cells
//        cell.textLabel?.text = tweet.text
//        cell.detailTextLabel?.text = tweet.user.name

        // our outlets to our custom UI
        // are connected to this custom UITableViewCell-subclassed cell
        // so we need to tell it which tweet is shown in its row
        // and it can load up its UI through its outlets
        if let tweetCell = cell as? TweetTableViewCell {
            tweetCell.tweet = tweet
        }

        return cell
    }

    // Added after lecture for REFRESHING
    override func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
        // make it a little clearer when each pull from Twitter
        // occurs in the table by setting section header titles
        return "\(tweets.count-section)"
    }
}
```

```
//
//  TweetTableViewCell.swift
//  Smashtag
//
//  Created by CS193p Instructor.
//  Copyright © 2017 Stanford University. All rights reserved.
//

import UIKit
import Twitter

class TweetTableViewCell: UITableViewCell
{
    // outlets to the UI components in our Custom UITableViewCell
    @IBOutlet weak var tweetProfileImageView: UIImageView!
    @IBOutlet weak var tweetCreatedLabel: UILabel!
    @IBOutlet weak var tweetUserLabel: UILabel!
    @IBOutlet weak var tweetTextLabel: UILabel!

    // public API of this UITableViewCell subclass
    // each row in the table has its own instance of this class
    // and each instance will have its own tweet to show
    // as set by this var
    var tweet: Twitter.Tweet? { didSet { updateUI() } }

    // whenever our public API tweet is set
    // we just update our outlets using this method
    private func updateUI() {
        tweetTextLabel?.text = tweet?.text
        tweetUserLabel?.text = tweet?.user.description

        if let profileImageURL = tweet?.user.profileImageURL {
            // FIXME: blocks main thread
            if let imageData = try? Data(contentsOf: profileImageURL) {
                tweetProfileImageView?.image = UIImage(data: imageData)
            }
        } else {
            tweetProfileImageView?.image = nil
        }

        if let created = tweet?.created {
            let formatter = DateFormatter()
            if Date().timeIntervalSince(created) > 24*60*60 {
                formatter.dateStyle = .short
            } else {
                formatter.timeStyle = .short
            }
            tweetCreatedLabel?.text = formatter.string(from: created)
        } else {
            tweetCreatedLabel?.text = nil
        }
    }
}
```