Background Fetch

This is a shortened version from <http://www.appcoda.com/ios7-background-fetch-programming/> released March 9th. The goal is to get an example of an app that stores current data locally and only updates when new and different data is available. This also shows how IOS 7 uses background fetch to repetitively check for new content while an app runs in the background.

**The Interface**

1. We’ll need a UITable View and a Toolbar.

2. Position the toolbar on top. The UiTable below

3. Make the Bar Button Identifier “Trash”

4. Add a UITableViewCell object to the table view.

5. Set its style to Subtitle in the attributes inspector

6. Set its Identifier to idCellNewsTitle

7. Make the title System 15 Bold and set the lines value to 3.

**Properties, Outlets and Actions**

1. Create an outlet for our UITableView name tblNews

2. Create an Action for our Trash can named removeDataFile

3 . Add the following delegate: <UITableViewDelegate, UITableViewDataSource>

to the ViewController.h (@interface ViewController : UIViewController <UITableViewDelegate, UITableViewDataSource>)

4. Add the following to the Private Section of the class (ViewController.m)

#import "XMLParser.h"

#define NewsFeed @"http://feeds.reuters.com/reuters/technologyNews"

@interface ViewController ()

@property (nonatomic, strong) UIRefreshControl \*refreshControl;

@property (nonatomic, strong) NSArray \*arrNewsData;

@property (nonatomic, strong) NSString \*dataFilePath;

-(void)refreshData;

-(void)performNewFetchedDataActionsWithDataArray:(NSArray \*)dataArray;

**Methods**

**ViewDidLoadMethod**

// 1. Make self the delegate and datasource of the table view.

[self.tblNews setDelegate:self];

[self.tblNews setDataSource:self];

// 2. Specify the data storage file path.

NSArray \*paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

NSString \*docDirectory = [paths objectAtIndex:0];

self.dataFilePath = [docDirectory stringByAppendingPathComponent:@"newsdata"];

// 3. Initialize the refresh control.

self.refreshControl = [[UIRefreshControl alloc] init];

[self.refreshControl addTarget:self

action:@selector(refreshData)

forControlEvents:UIControlEventValueChanged];

[self.tblNews addSubview:self.refreshControl];

**Refreshing the Data:**

**refreshData Method**

-(void)refreshData{

XMLParser \*xmlParser = [[XMLParser alloc] initWithXMLURLString:NewsFeed];

[xmlParser startParsingWithCompletionHandler:^(BOOL success, NSArray \*dataArray, NSError \*error) {

if (success) {

[self performNewFetchedDataActionsWithDataArray:dataArray];

[self.refreshControl endRefreshing];

}

else{

NSLog(@"%@", [error localizedDescription]);

}

}];

}

**Doing Something with the data**

-(void)performNewFetchedDataActionsWithDataArray:(NSArray \*)dataArray{

**// 1. Initialize the arrNewsData array with the parsed data array.**

if (self.arrNewsData != nil) {

self.arrNewsData = nil;

}

self.arrNewsData = [[NSArray alloc] initWithArray:dataArray];

**// 2. Reload the table view.**

[self.tblNews reloadData];

**// 3. Save the data permanently to file.**

if (![self.arrNewsData writeToFile:self.dataFilePath atomically:YES]) {

NSLog(@"Couldn't save data.");

}

}

**Displaying the Data**

-(NSInteger)numberOfSectionsInTableView:(UITableView \*)tableView{

return 1;

}

-(NSInteger)tableView:(UITableView \*)tableView numberOfRowsInSection:(NSInteger)section{

return self.arrNewsData.count;

}

-(UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath{

UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:@"idCellNewsTitle"];

if (cell == nil) {

cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:@"idCellNewsTitle"];

}

NSDictionary \*dict = [self.arrNewsData objectAtIndex:indexPath.row];

cell.textLabel.text = [dict objectForKey:@"title"];

cell.detailTextLabel.text = [dict objectForKey:@"pubDate"];

return cell;

}

-(CGFloat)tableView:(UITableView \*)tableView heightForRowAtIndexPath:(NSIndexPath \*)indexPath{

return 80.0;

}

-(void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath{

NSDictionary \*dict = [self.arrNewsData objectAtIndex:indexPath.row];

NSString \*newsLink = [dict objectForKey:@"link"];

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:newsLink]];

}

**Data Loading**

**Add to the end of the ViewDidLoad Method:**

if ([[NSFileManager defaultManager] fileExistsAtPath:self.dataFilePath]) {

self.arrNewsData = [[NSMutableArray alloc] initWithContentsOfFile:self.dataFilePath];

[self.tblNews reloadData];

}

**Now to make the background work**

1. Click on the Project Navigator
2. Click on Capabilities
3. Set Background Modes to On
4. Check Background Fetch

**Now for our AppDelegate.m**

**Add #import "ViewController.h"**

**Add this to our didFinishLaunchingWithOptions method:**

[application setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum];

**//this lets the system decide how frequently to check for new data.**

**Now add the following completion handler:**

-(void)application:(UIApplication \*)application performFetchWithCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler{

NSDate \*fetchStart = [NSDate date];

//now you know why we imported the ViewController.h in our appdelegate.

ViewController \*viewController = (ViewController \*)self.window.rootViewController;

[viewController fetchNewDataWithCompletionHandler:^(UIBackgroundFetchResult result) {

completionHandler(result);

//**Let’s track the time:**

NSDate \*fetchEnd = [NSDate date];

NSTimeInterval timeElapsed = [fetchEnd timeIntervalSinceDate:fetchStart];

NSLog(@"Background Fetch Duration: %f seconds", timeElapsed);

}];

}

**Let’s delete the data (clicking the trash icon).**

- (IBAction)removeDataFile:(id)sender {

if ([[NSFileManager defaultManager] fileExistsAtPath:self.dataFilePath]) {

[[NSFileManager defaultManager] removeItemAtPath:self.dataFilePath error:nil];

self.arrNewsData = nil;

[self.tblNews reloadData];

}

}