Developing Applications for iOS

Lab 6:
Nearby Deals (2 of 6)

Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Nearby Deals

Description:

We are going to build a new application that will show deals from nearby restaurants and bars. The application will display the deals in two modes: a list view (using a `UITableViewController`) and a map view (using a `MKMapView`). We will request the deals from a server (www.geoadsplus.com to be more precise). We will use XML to communicate with this server. Note that XML and JSON are standard ways of communicating with a server.

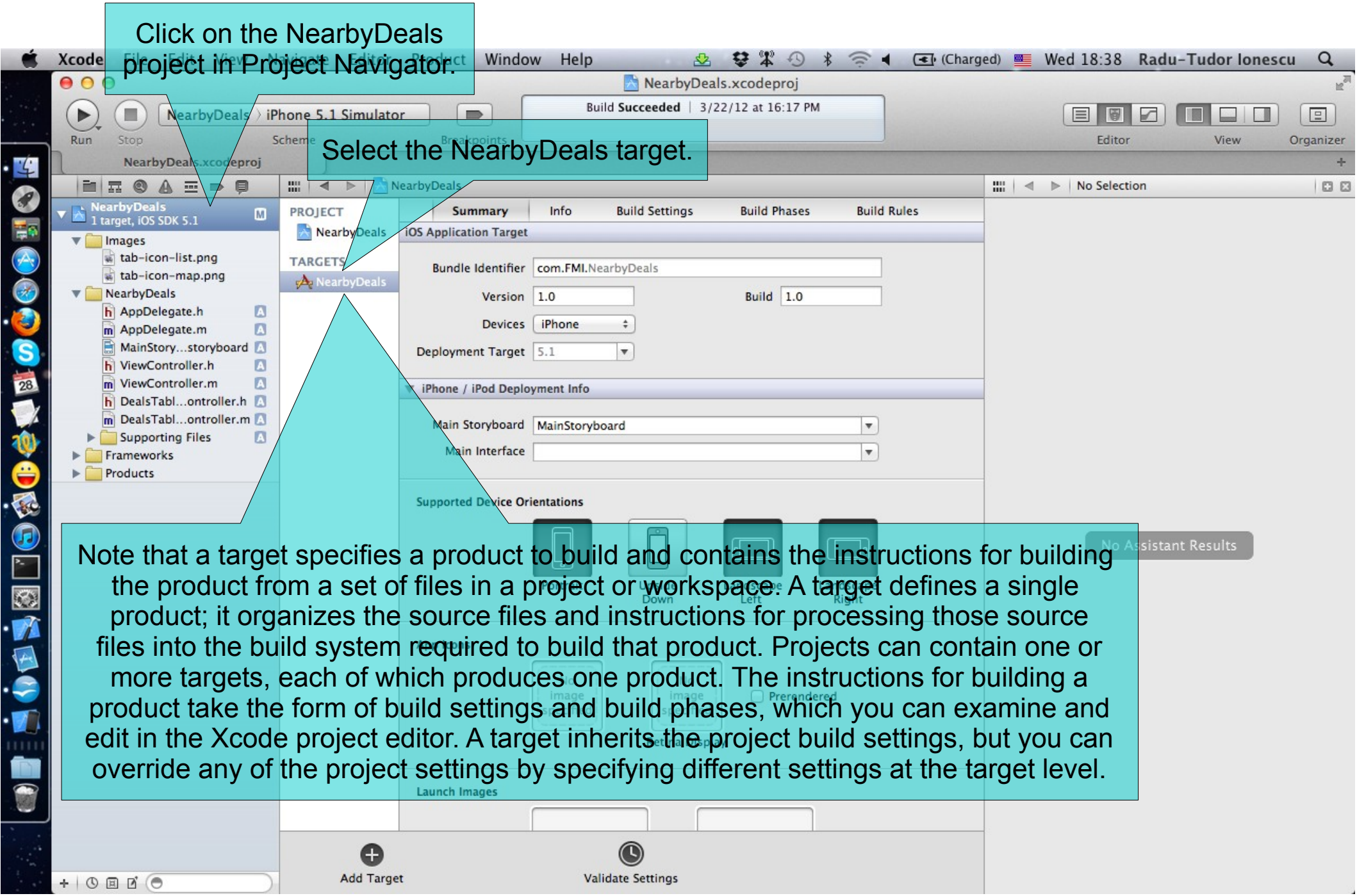
We have to pass the device location (latitude, longitude) to the server so that it gives us nearby deals. Thus, we will need to use Location Services to determine the device location.

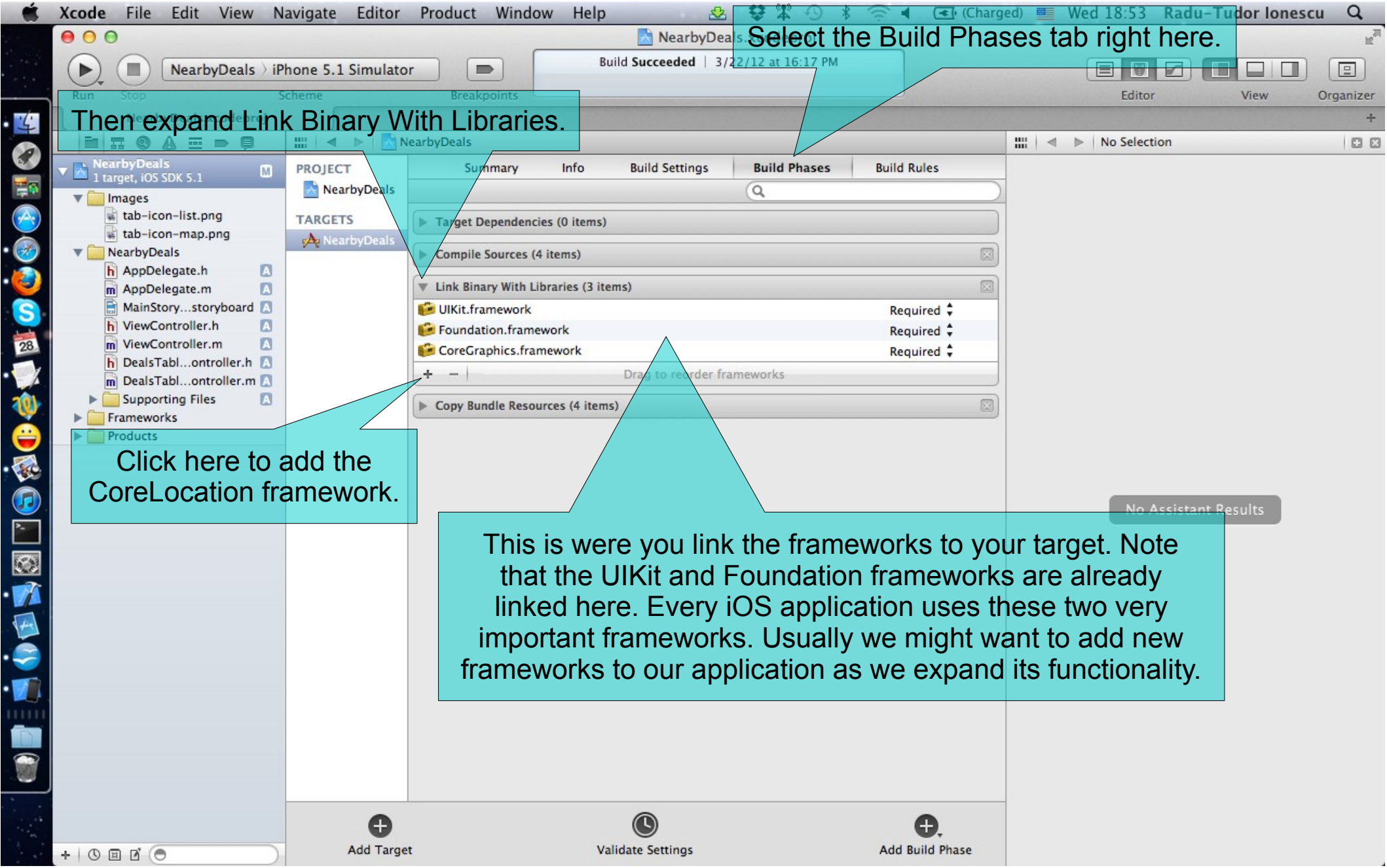
We will offer details about our deals. We are going to use a navigation controller to navigate between the list View and the details View.

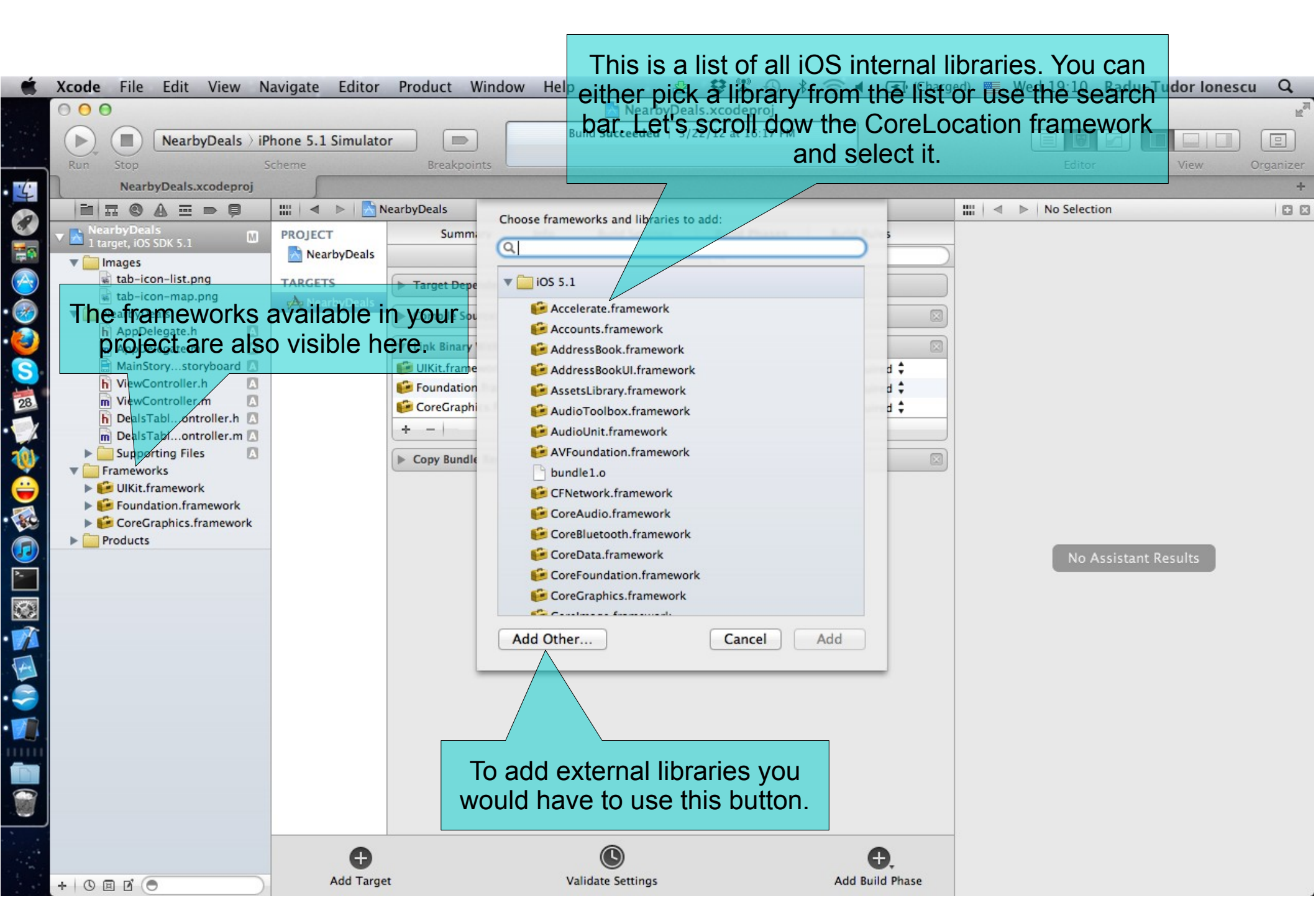
Task 1

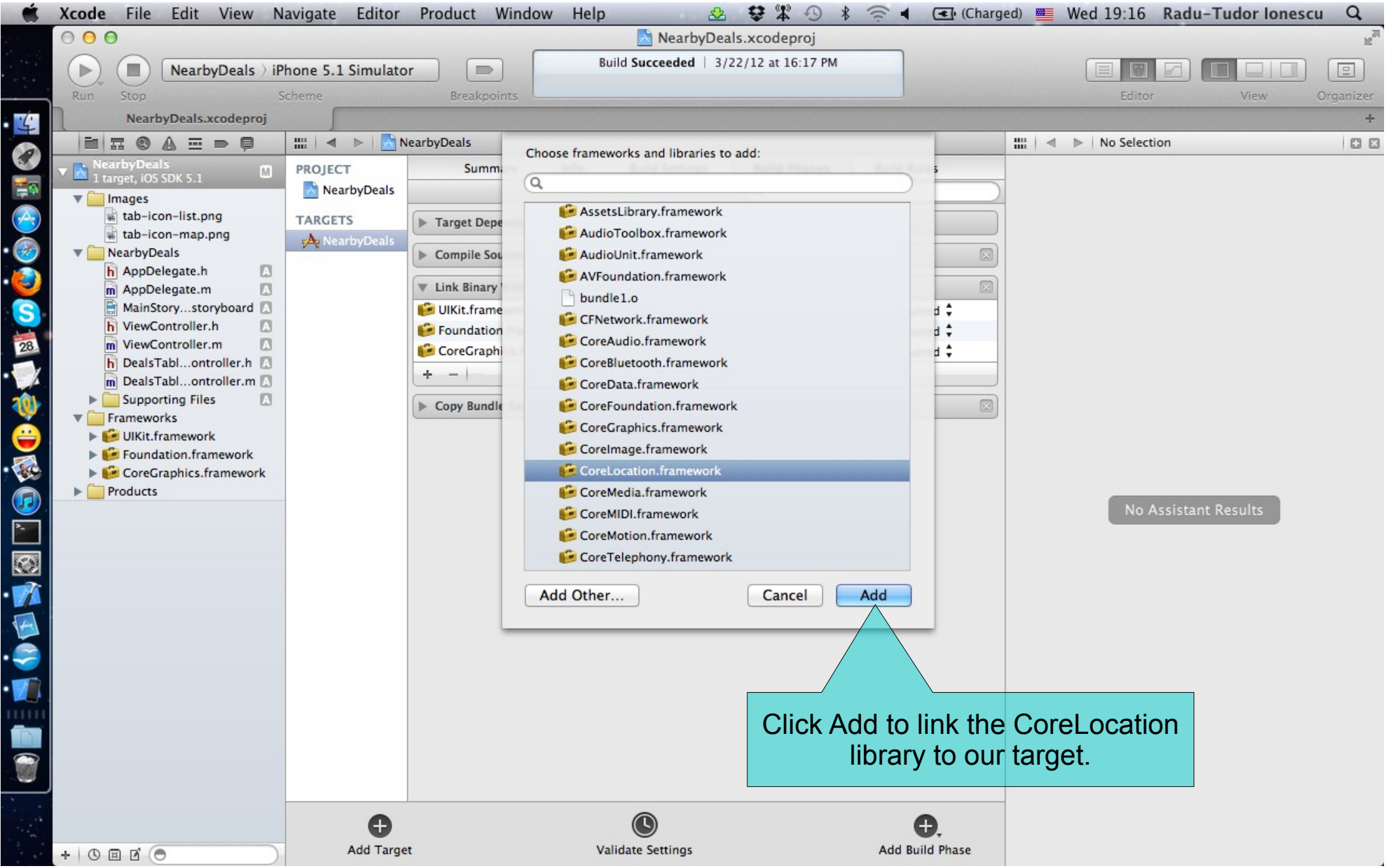
Task: Add the CoreLocation framework to your project.

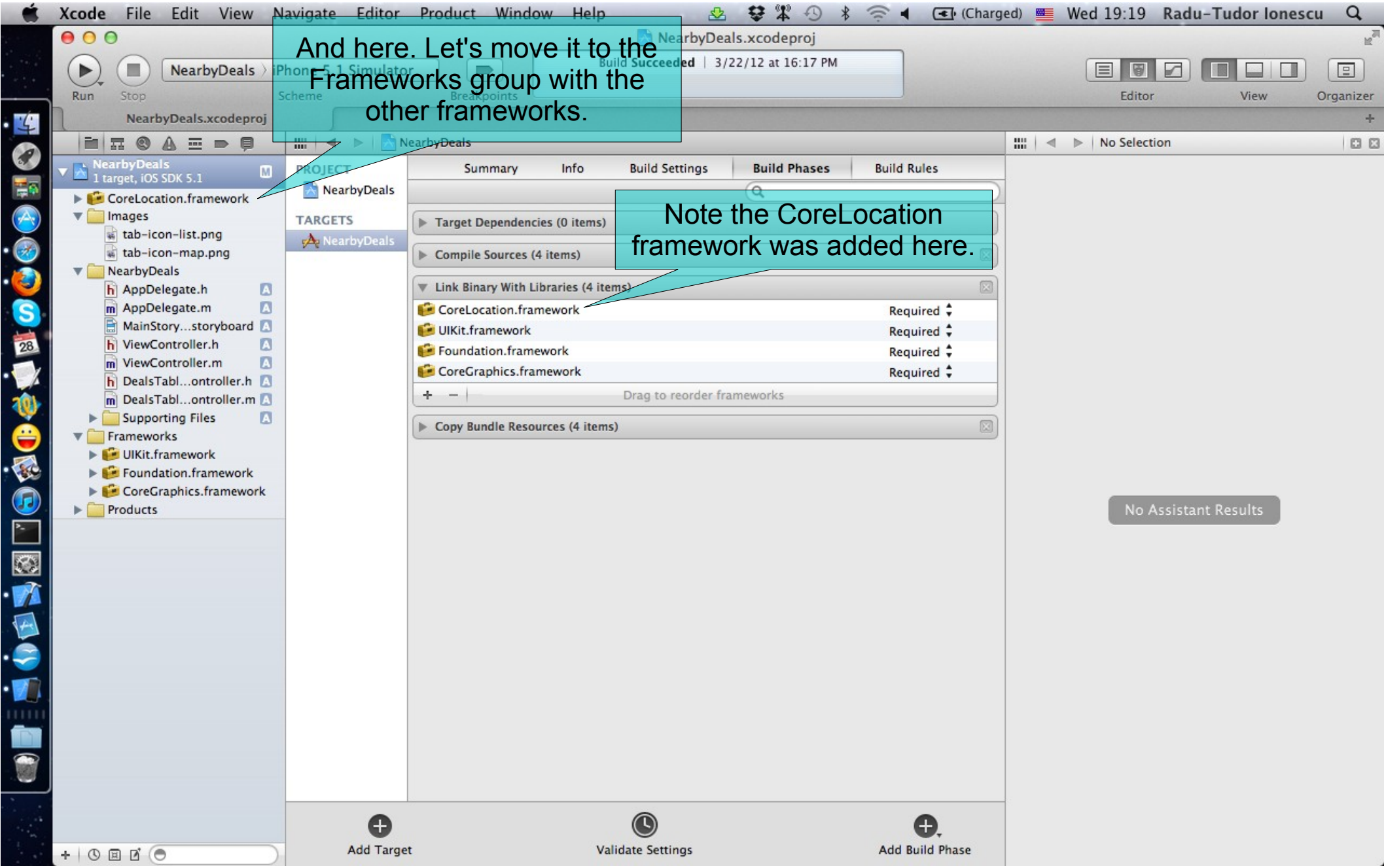
1. Launch Xcode and go to “File > Open” and select the Xcode project (.xcodeproj) inside the “NearbyDeals(1of6)” folder. You can also double-click on the .xcodeproj file to open it in Xcode.
2. Run the application in iOS Simulator and take a look to remember what was done last time.
3. Stop running the application.
4. Open Project Navigator and follow the steps in the next slides to add the CoreLocation framework to your project.

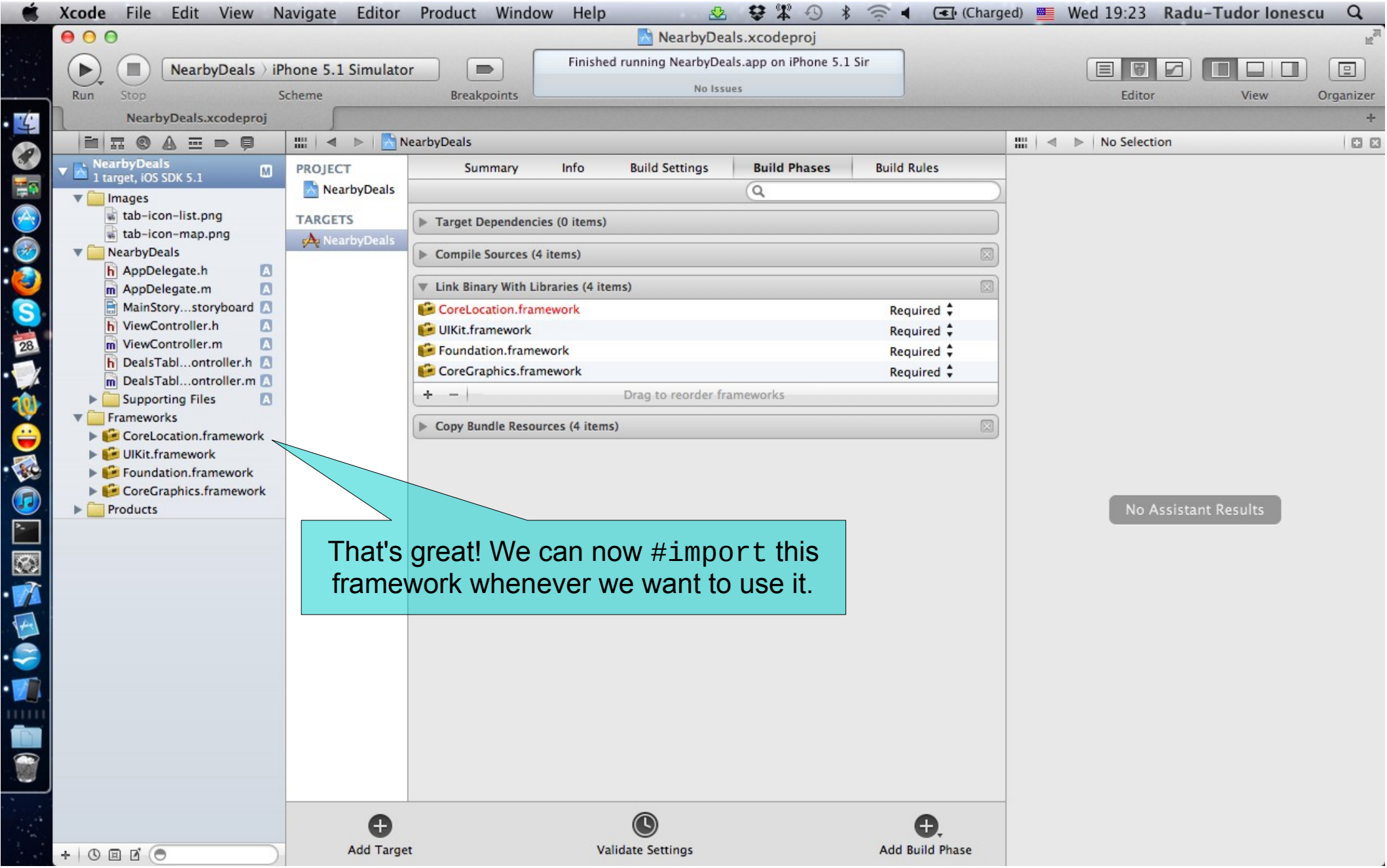

Click on the NearbyDeals project in Project Navigator.
Select the NearbyDeals target.
Note that a target specifies a product to build and contains the instructions for building the product from a set of files in a project or workspace. A target defines a single product; it organizes the source files and instructions for processing those source files into the build system required to build that product. Projects can contain one or more targets, each of which produces one product. The instructions for building a product take the form of build settings and build phases, which you can examine and edit in the Xcode project editor. A target inherits the project build settings, but you can override any of the project settings by specifying different settings at the target level.
Xcode
Window
Help
Wed 18:38
Radu-Tudor Ionescu
NearbyDeals.xcodeproj
Build Succeeded | 3/22/12 at 16:17 PM
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
NearbyDeals
1 target, iOS SDK 5.1
Images
tab-icon-list.png
tab-icon-map.png
NearbyDeals
AppDelegate.h
AppDelegate.m
MainStory...storyboard
ViewController.h
ViewController.m
DealsTabl...ontroller.h
DealsTabl...ontroller.m
Supporting Files
Frameworks
Products
PROJECT
NearbyDeals
TARGETS
NearbyDeals
Summary
Info
Build Settings
Build Phases
Build Rules
iOS Application Target
Bundle Identifier
com.FMI.NearbyDeals
Version
1.0
Build
1.0
Devices
iPhone
Deployment Target
5.1
iPhone / iPod Deployment Info
Main Storyboard
MainStoryboard
Main Interface
Supported Device Orientations
Launch Images
No Selection
No Assistant Results
Add Target
Validate Settings


Select the Build Phases tab right here.
Then expand Link Binary With Libraries.
Click here to add the CoreLocation framework.
This is were you link the frameworks to your target. Note that the UIKit and Foundation frameworks are already linked here. Every iOS application uses these two very important frameworks. Usually we might want to add new frameworks to our application as we expand its functionality.
Summary
Info
Build Settings
Build Phases
Build Rules
Target Dependencies (0 items)
Compile Sources (4 items)
Link Binary With Libraries (3 items)
UIKit.framework
Foundation.framework
CoreGraphics.framework
Required
Copy Bundle Resources (4 items)
Add Target
Validate Settings
Add Build Phase


This is a list of all iOS internal libraries. You can either pick a library from the list or use the search bar. Let's scroll dow the CoreLocation framework and select it.
The frameworks available in your project are also visible here.
To add external libraries you would have to use this button.
Choose frameworks and libraries to add:
iOS 5.1
Accelerate.framework
Accounts.framework
AddressBook.framework
AddressBookUI.framework
AssetsLibrary.framework
AudioToolbox.framework
AudioUnit.framework
AVFoundation.framework
bundle1.o
CFNetwork.framework
CoreAudio.framework
CoreBluetooth.framework
CoreData.framework
CoreFoundation.framework
CoreGraphics.framework
Add Other...
Cancel
Add
No Assistant Results
Add Target
Validate Settings
Add Build Phase


Choose frameworks and libraries to add:
AssetsLibrary.framework
AudioToolbox.framework
AudioUnit.framework
AVFoundation.framework
bundle1.o
CFNetwork.framework
CoreAudio.framework
CoreBluetooth.framework
CoreData.framework
CoreFoundation.framework
CoreGraphics.framework
CoreImage.framework
CoreLocation.framework
CoreMedia.framework
CoreMIDI.framework
CoreMotion.framework
CoreTelephony.framework
Add Other...
Cancel
Add
Click Add to link the CoreLocation library to our target.
No Assistant Results
Add Target
Validate Settings
Add Build Phase


And here. Let's move it to the Frameworks group with the other frameworks.
Note the CoreLocation framework was added here.
Xcode File Edit View Navigate Editor Product Window Help
Wed 19:19 Radu-Tudor Ionescu
NearbyDeals.xcodeproj
Build Succeeded | 3/22/12 at 16:17 PM
Run Stop Scheme Breakpoints Editor View Organizer
NearbyDeals
1 target, iOS SDK 5.1
CoreLocation.framework
Images
tab-icon-list.png
tab-icon-map.png
NearbyDeals
AppDelegate.h
AppDelegate.m
MainStory...storyboard
ViewController.h
ViewController.m
DealsTabl...ontroller.h
DealsTabl...ontroller.m
Supporting Files
Frameworks
UIKit.framework
Foundation.framework
CoreGraphics.framework
Products
PROJECT
NearbyDeals
TARGETS
NearbyDeals
Summary Info Build Settings Build Phases Build Rules
Target Dependencies (0 items)
Compile Sources (4 items)
Link Binary With Libraries (4 items)
CoreLocation.framework Required
UIKit.framework Required
Foundation.framework Required
CoreGraphics.framework Required
Drag to reorder frameworks
Copy Bundle Resources (4 items)
Add Target
Validate Settings
Add Build Phase
No Selection
No Assistant Results


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Wed 19:23 Radu-Tudor Ionescu
NearbyDeals.xcodeproj
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Finished running NearbyDeals.app on iPhone 5.1 Sir
No Issues
Editor View Organizer
NearbyDeals.xcodeproj
NearbyDeals
No Selection
NearbyDeals
1 target, iOS SDK 5.1
Images
tab-icon-list.png
tab-icon-map.png
NearbyDeals
AppDelegate.h
AppDelegate.m
MainStory...storyboard
ViewController.h
ViewController.m
DealsTabl...ontroller.h
DealsTabl...ontroller.m
Supporting Files
Frameworks
CoreLocation.framework
UIKit.framework
Foundation.framework
CoreGraphics.framework
Products
PROJECT
NearbyDeals
TARGETS
NearbyDeals
Summary Info Build Settings Build Phases Build Rules
Target Dependencies (0 items)
Compile Sources (4 items)
Link Binary With Libraries (4 items)
CoreLocation.framework Required
UIKit.framework Required
Foundation.framework Required
CoreGraphics.framework Required
Drag to reorder frameworks
Copy Bundle Resources (4 items)
No Assistant Results
That's great! We can now #import this framework whenever we want to use it.
Add Target
Validate Settings
Add Build Phase

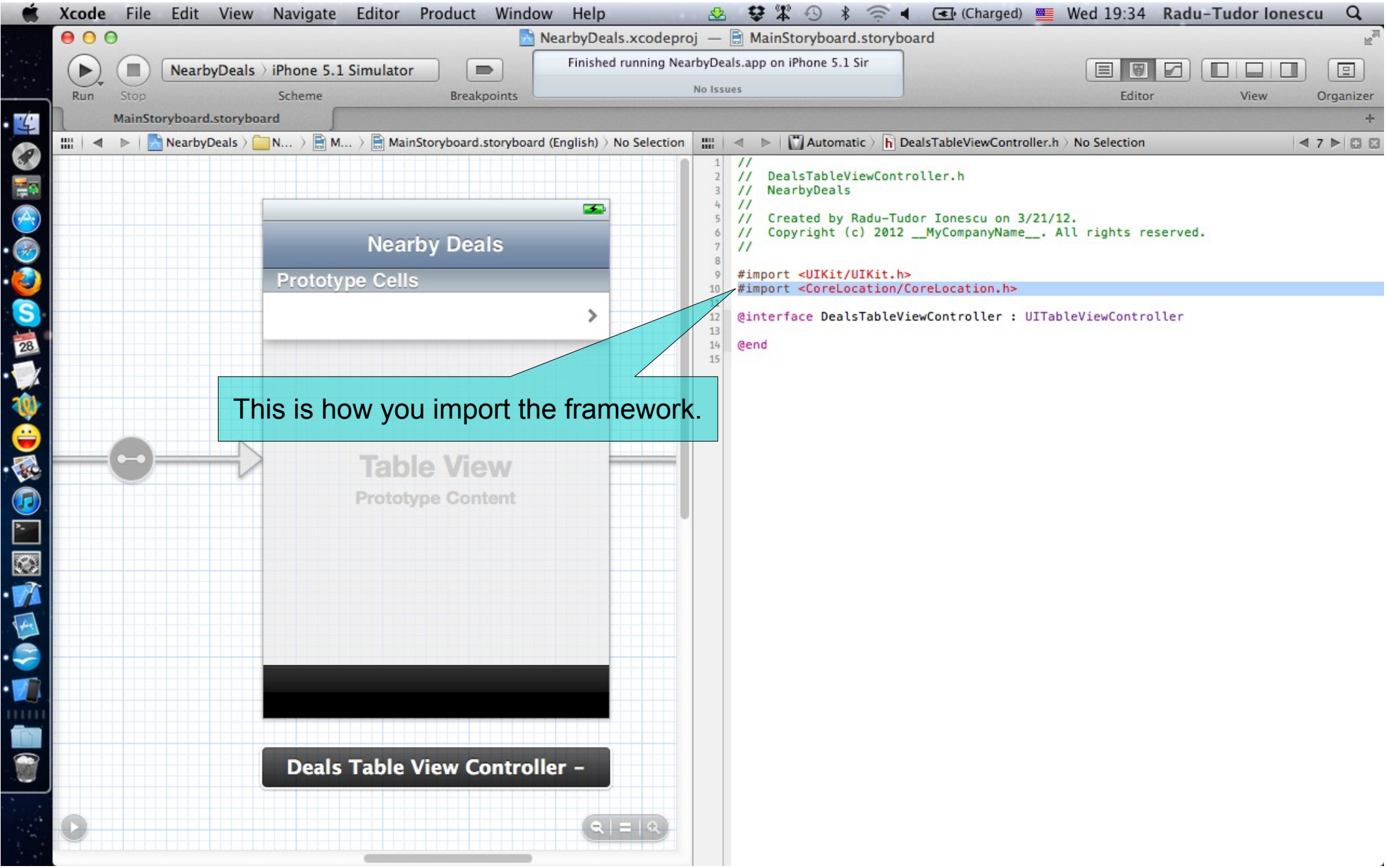
Task 1

Task: Add the CoreLocation framework to your project.

5. Select the MainStoryboard.storyboard file.

6. Hide Project Navigator and let's `#import` the CoreLocation framework inside the DealsTableViewController.

We need this framework because we will declare a method soon that has a `CLLocationCoordinate2D` argument. This is a C struct from the CoreLocation library that contains two properties: `latitude` and `longitude`.


Xcode File Edit View Navigate Editor Product Window Help
Wed 19:34 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
NearbyDeals iPhone 5.1 Simulator
Finished running NearbyDeals.app on iPhone 5.1 Sir
No Issues
Editor View Organizer
MainStoryboard.storyboard
NearbyDeals N... M... MainStoryboard.storyboard (English) No Selection
Automatic DealsTableViewController.h No Selection
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
This is how you import the framework.
//
// DealsTableViewController.h
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 3/21/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>
@interface DealsTableViewController : UITableViewController
@end

Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

1. Before creating a request from code, we should check out the GeoAds+ API in a web browser. Let's open the API documentation first: http://www.geoadsplus.com/api_documentation in Safari.

2. Note that you need an app_key to make a request to the GeoAds+ API. We will use this one: fe008041973b66760017.

3. Let's try the following request in Safari and see what we get:

http://www.geoadsplus.com/ads.xml?
app_key=fe008041973b66760017&latitude=44.25&longitude=26.06
&limit=20&category=Restaurants,Bars

4. Note that you should right-click in Safari and select View Source to see the XML returned by the GeoAds+ API.

Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

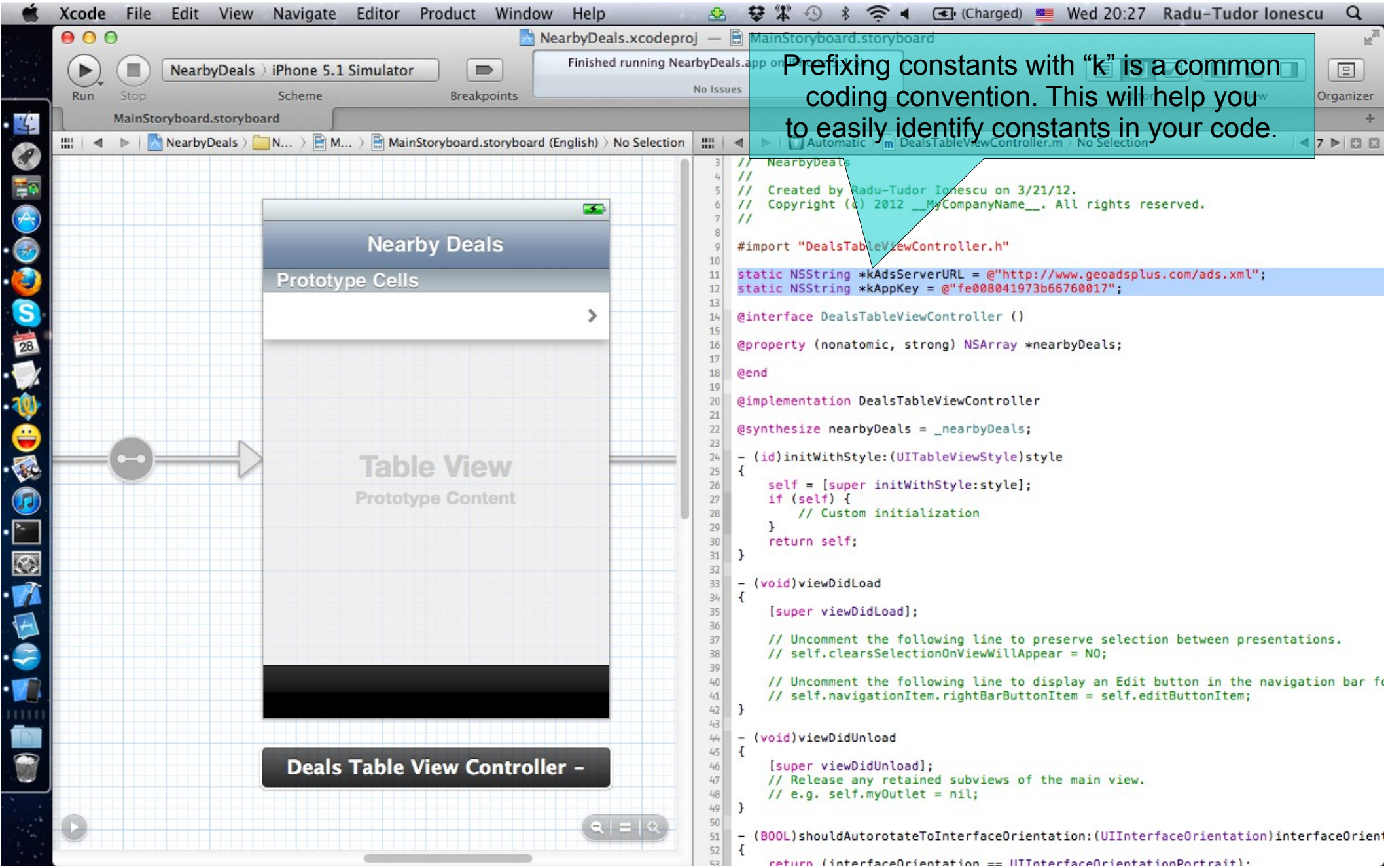
5. Let's return to our project in Xcode. We will declare two constant strings for the GeoAds+ URL and for the app key.

6. Select DealsTableViewController.m in Assistant Editor.

7. Right after the `#import` declare a `static NSString` that holds this URL: http://www.geoadsplus.com/ads.xml.

8. In a similar way, declare another `static NSString` to hold our app key: fe008041973b66760017.

The next screenshot shows you how to declare these string constants.


Prefixing constants with “k” is a common coding convention. This will help you to easily identify constants in your code.
Xcode File Edit View Navigate Editor Product Window Help
Wed 20:27 Radu-Tudor Ionescu
NearbyDeals.xcodeproj
MainStoryboard.storyboard
Run Stop Scheme Breakpoints Organizer
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
static NSString *kAdsServerURL = @"http://www.geoadsplus.com/ads.xml";
static NSString *kAppKey = @"fe008041973b66760017";

Task 2

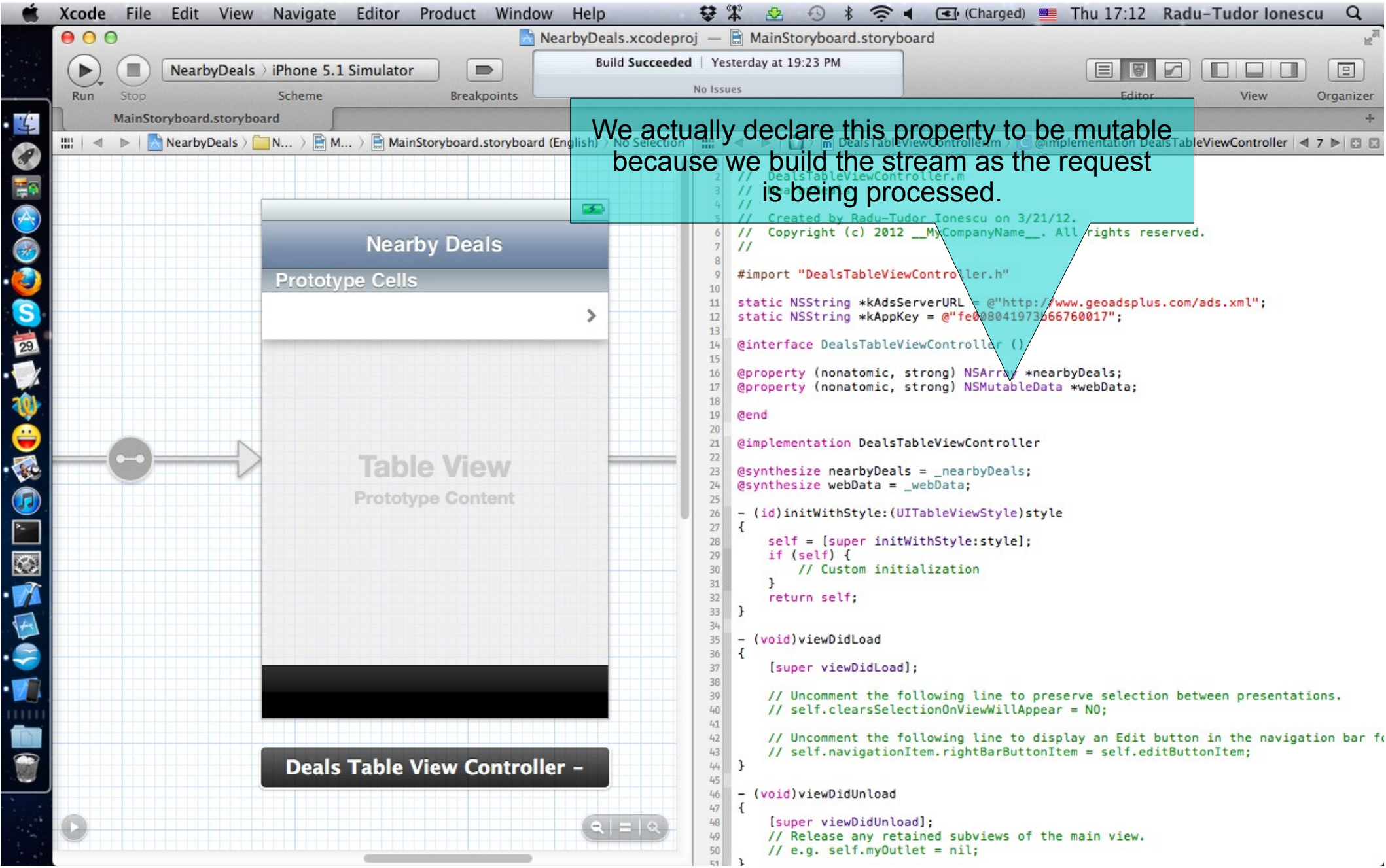
Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

9. Next, we are going to make an asynchronous request to the GeoAds+ API. For this, we implement a method that starts the request by using an `NSURLConnection` object. While this request is processed by the server it can send some messages to its delegate.

We will declare a private `@property` that will contain the stream of bytes (`NSData`) that the delegate receives through the `NSURLConnection` object. Name this property `webData`. Because we (the Controller) have the only reference to it, we use the `strong` storage type.

10. Synthesize this property and prefix its instance variable with underscore.

The next screenshot shows you how to do these steps.


We actually declare this property to be mutable because we build the stream as the request is being processed.
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –

Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

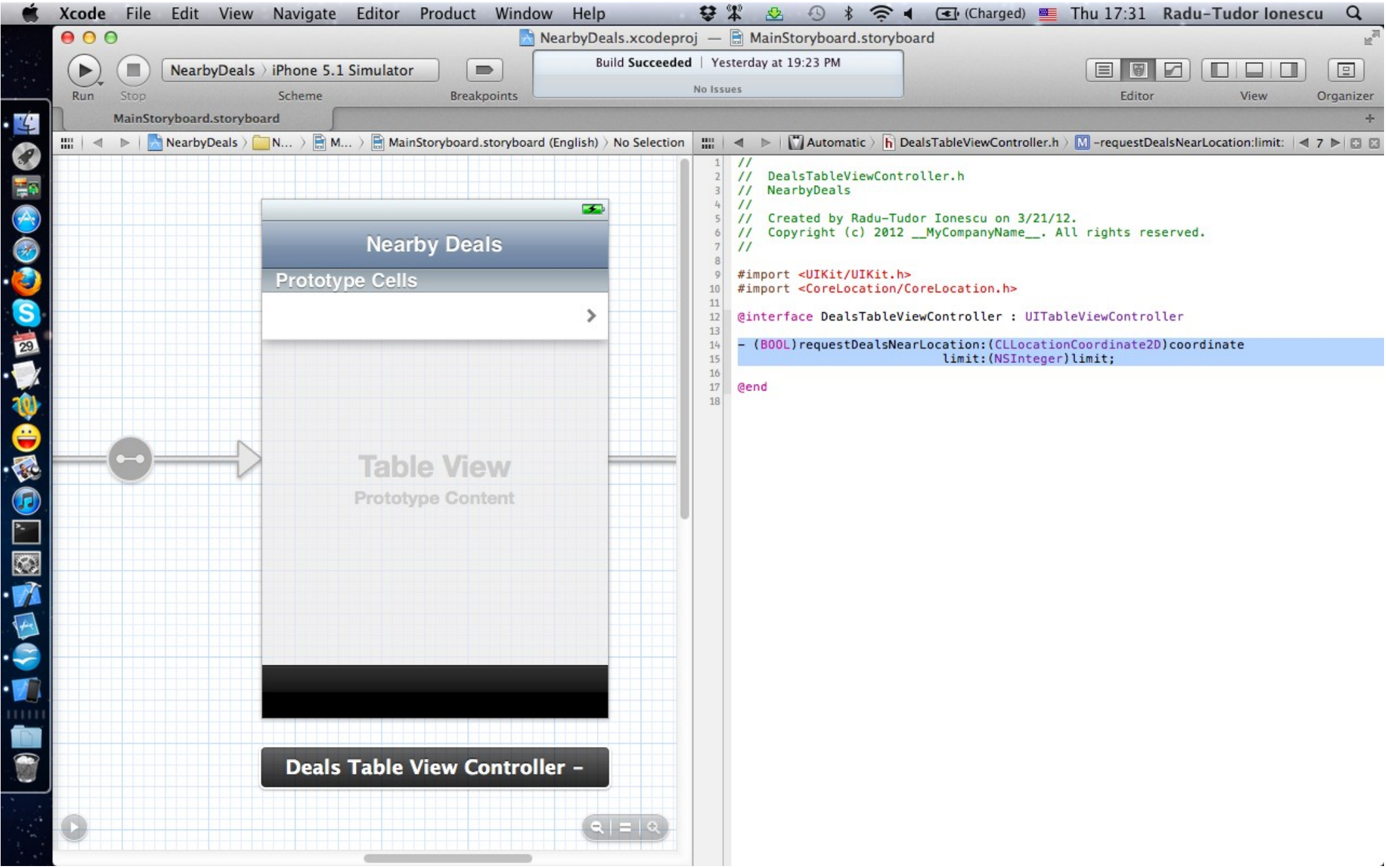
11. Let's declare and implement the method that makes the request for nearby deals to the GeoAds+ API.

Open DealsTableViewController.h in Assistant Editor.

12. Declare the `requestDealsNearLocation:limit:` method that has two arguments: the location (GPS coordinates) of the device and the maximum number of deals to be returned by the server.

This method will return a `BOOL` value that indicates whether the request was successfully started or not.

The next screenshot shows you how to do declare this method.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 17:31 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
NearbyDeals N... M... MainStoryboard.storyboard (English) No Selection
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
Automatic DealsTableViewController.h -requestDealsNearLocation:limit:
1 //
2 // DealsTableViewController.h
3 // NearbyDeals
4 //
5 // Created by Radu-Tudor Ionescu on 3/21/12.
6 // Copyright (c) 2012 __MyCompanyName__. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10 #import <CoreLocation/CoreLocation.h>
11
12 @interface DealsTableViewController : UITableViewController
13
14 - (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
15 limit:(NSInteger)limit;
16
17 @end
18

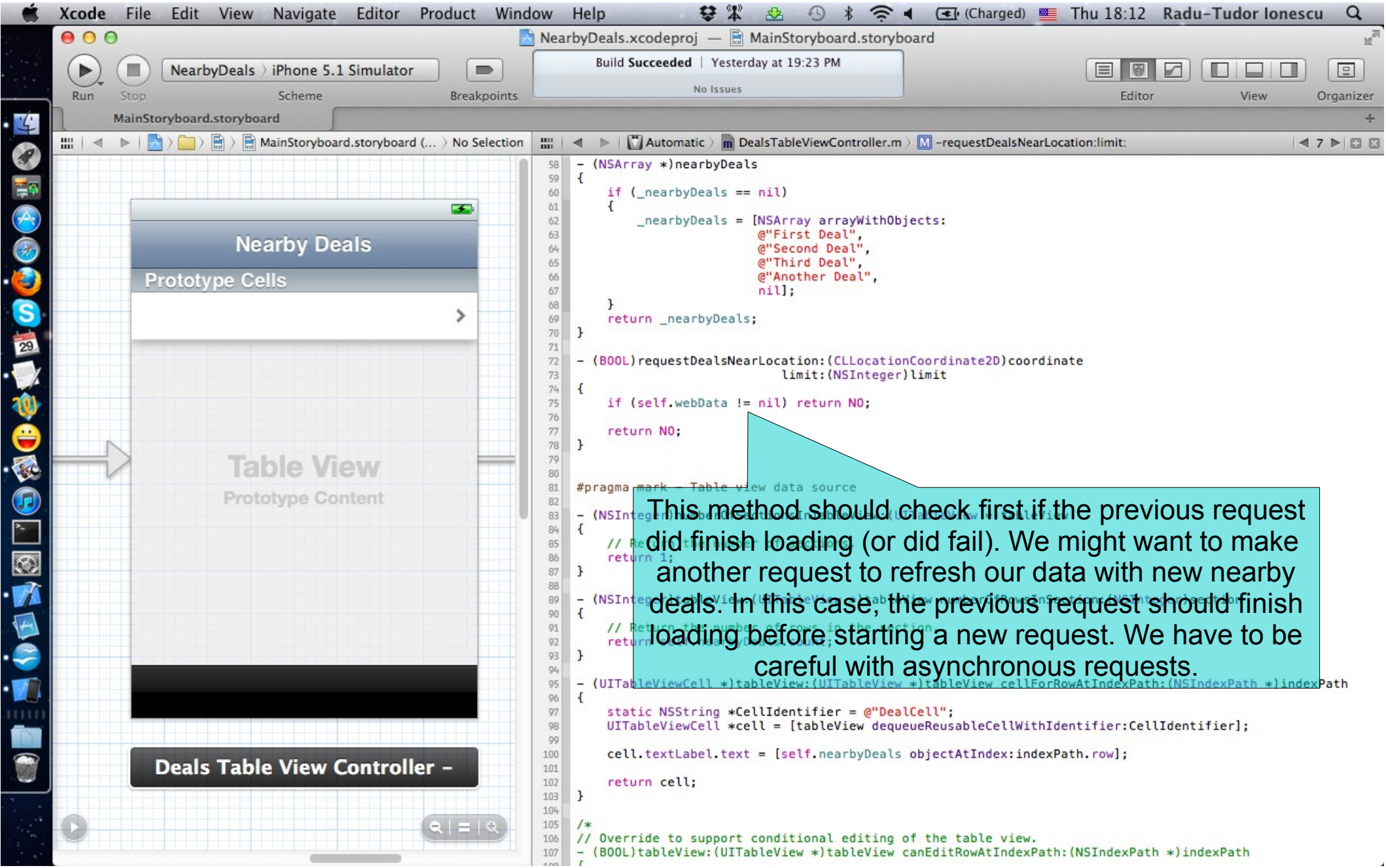
Task 2

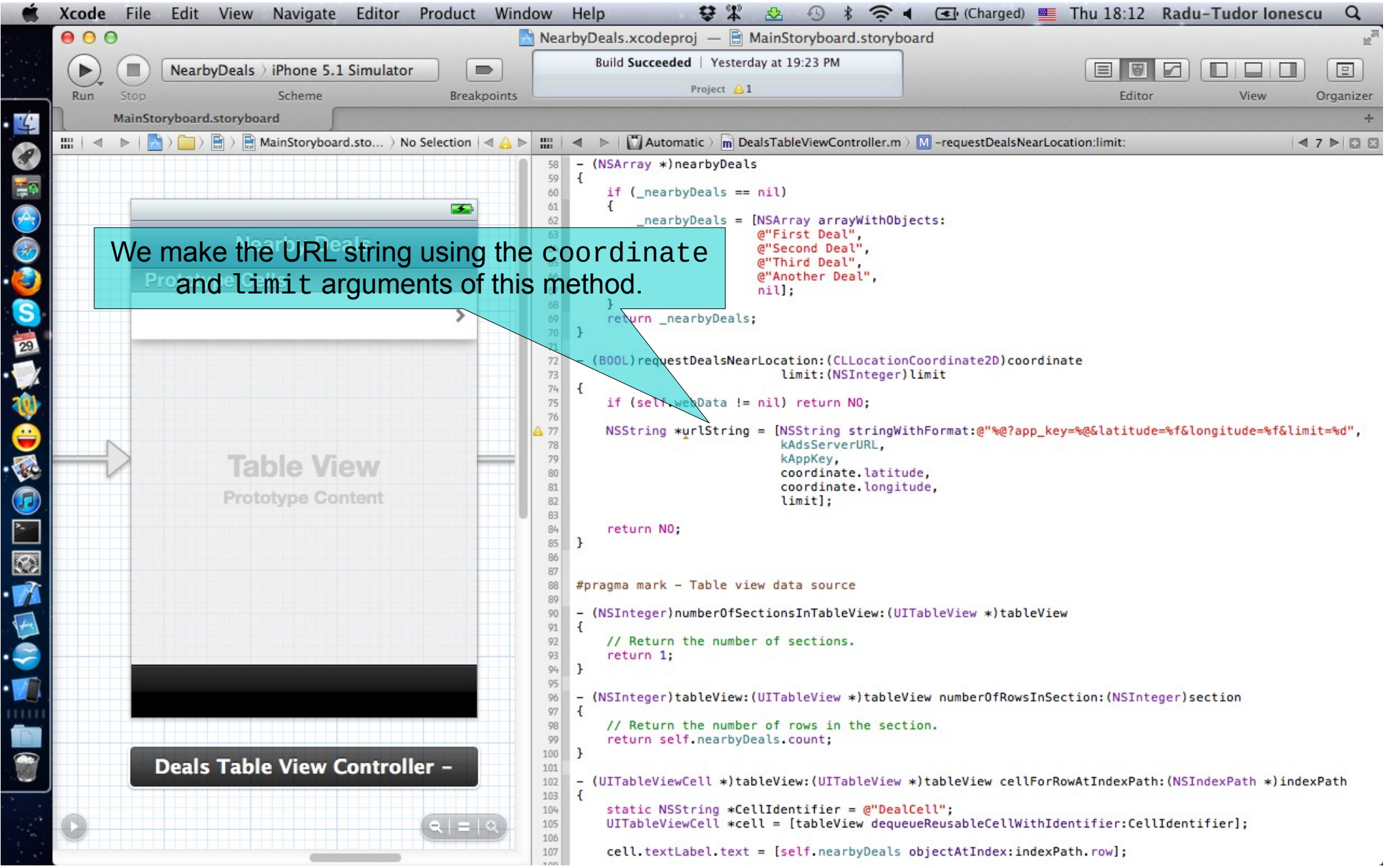
Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

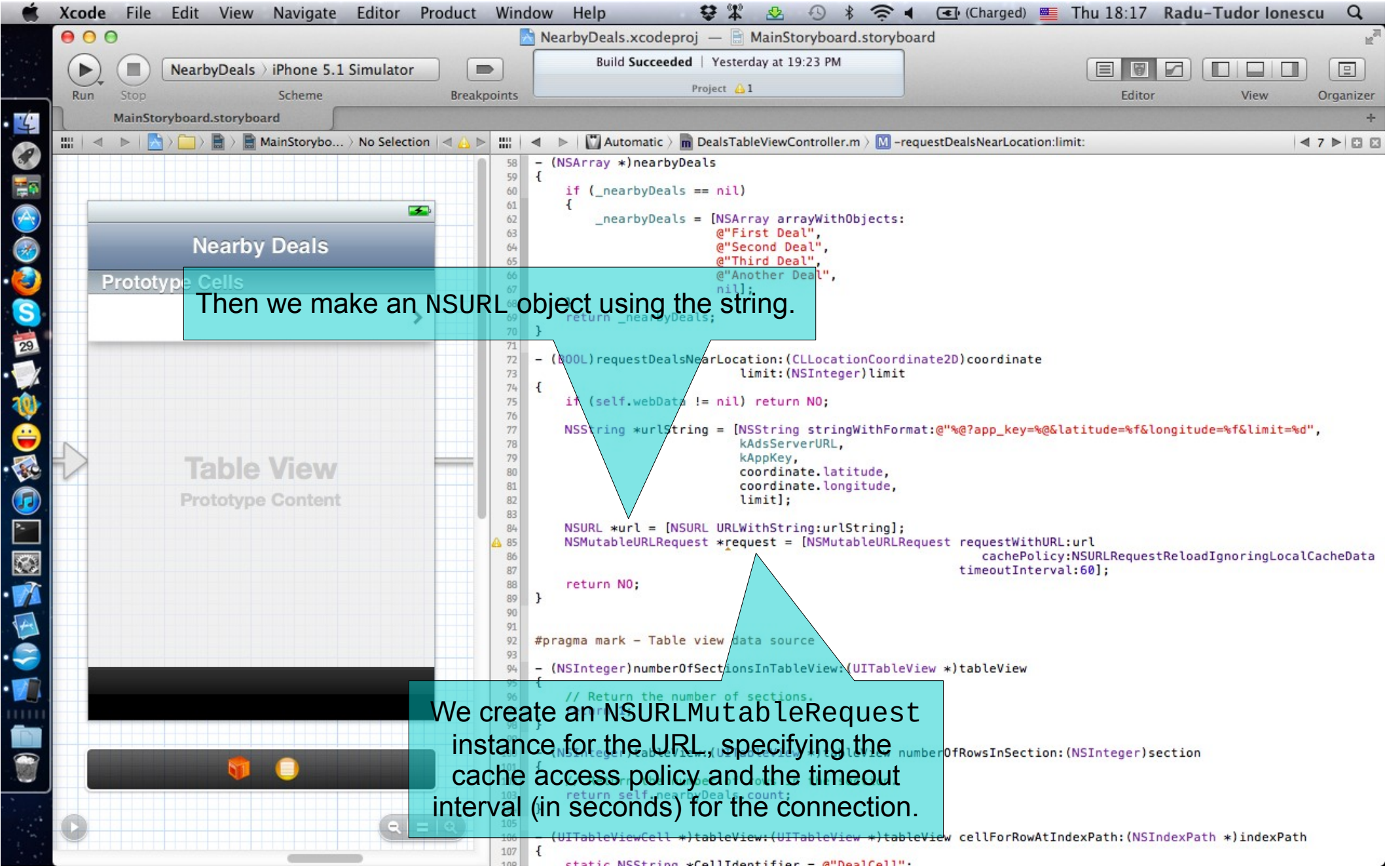
13. Go back to DealsTableViewController.m in Assistant Editor.

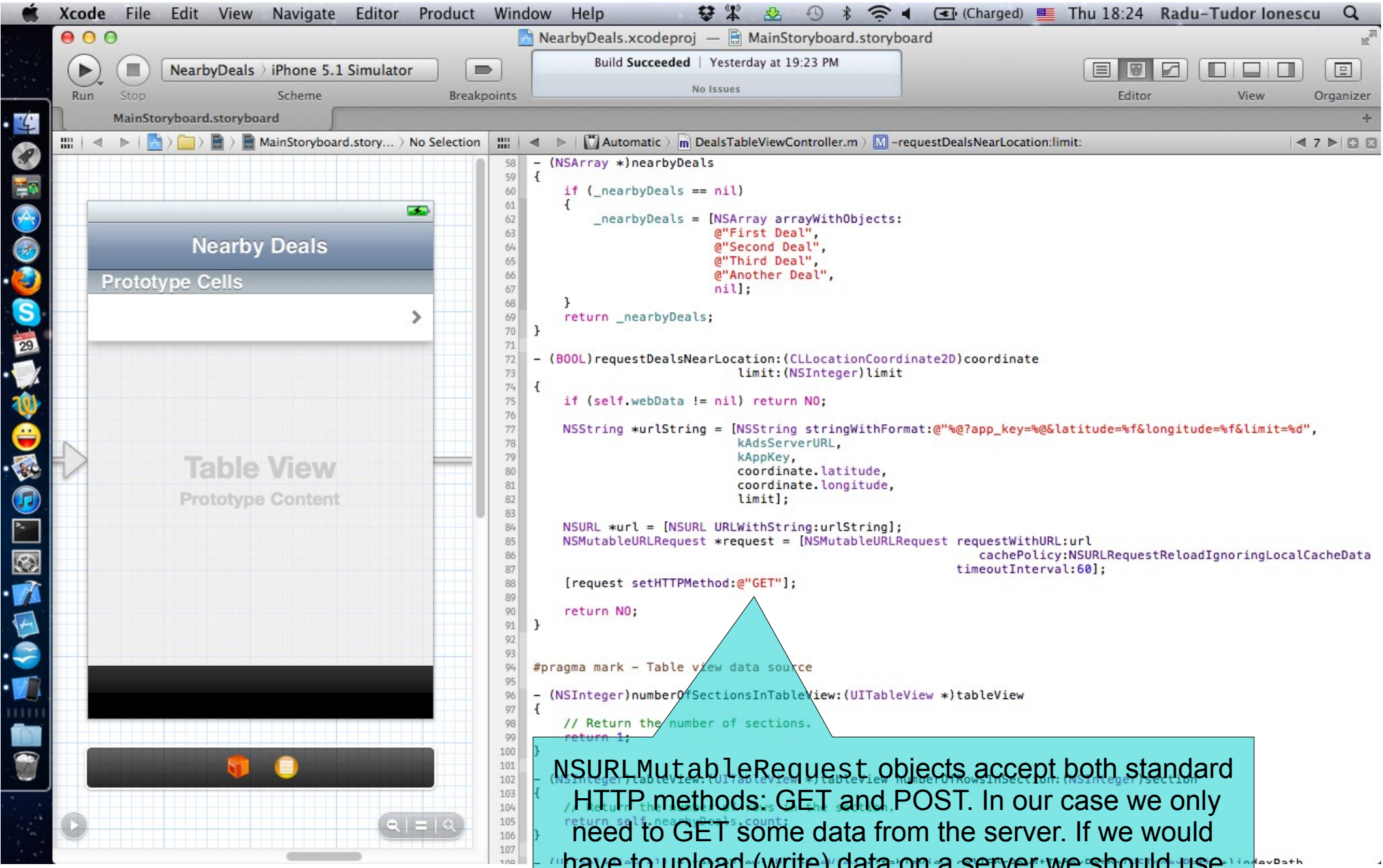
14. Implement the `requestDealsNearLocation:limit:` method after the `nearbyDeals` getter.

The next screenshots guide you through the implementation of this method.

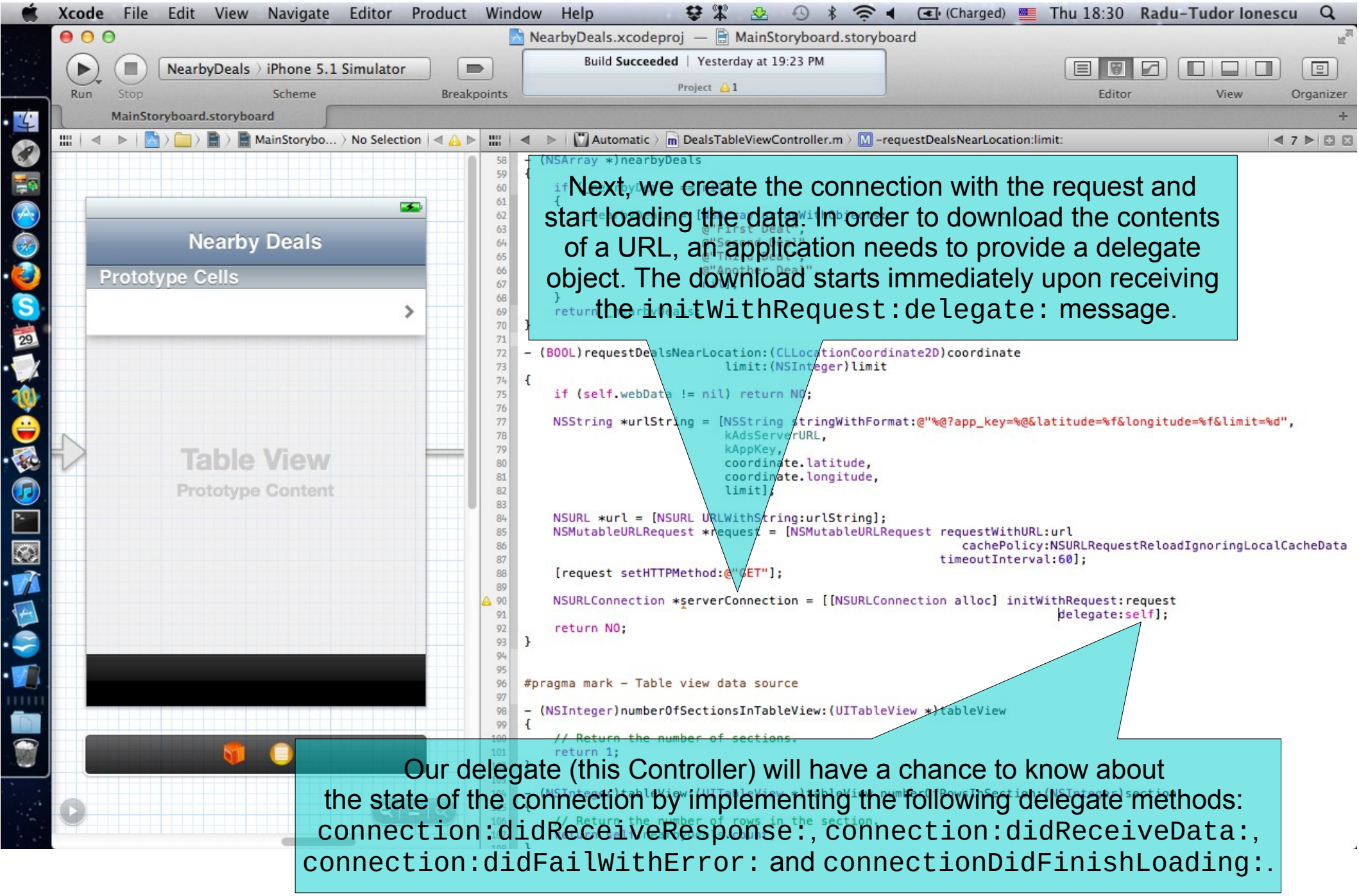

Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 18:12 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.storyboard (... No Selection
Automatic DealsTableViewController.m -requestDealsNearLocation:limit:
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
- (NSArray *)nearbyDeals
{
if (_nearbyDeals == nil)
{
_nearbyDeals = [NSArray arrayWithObjects:
@"First Deal",
@"Second Deal",
@"Third Deal",
@"Another Deal",
nil];
}
return _nearbyDeals;
}
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit
{
if (self.webData != nil) return NO;
return NO;
}
#pragma mark - Table view data source
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"DealCell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
cell.textLabel.text = [self.nearbyDeals objectAtIndex:indexPath.row];
return cell;
}
/*
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
This method should check first if the previous request did finish loading (or did fail). We might want to make another request to refresh our data with new nearby deals. In this case, the previous request should finish loading before starting a new request. We have to be careful with asynchronous requests.

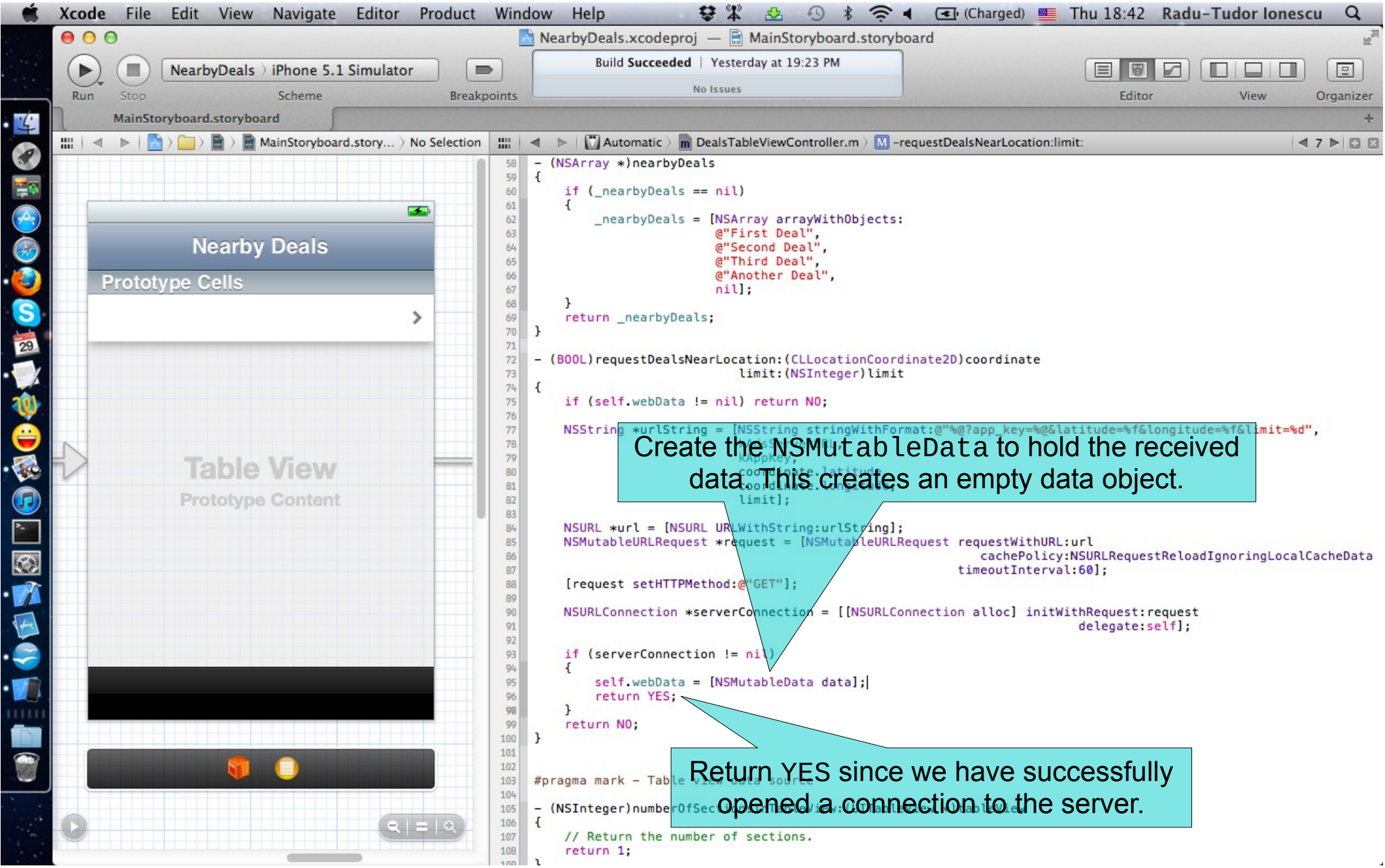

We make the URL string using the coordinate and limit arguments of this method.
Table View
Prototype Content
Deals Table View Controller –


Then we make an NSURL object using the string.
We create an NSURLMutableRequest instance for the URL, specifying the cache access policy and the timeout interval (in seconds) for the connection.



NSURLMutableRequest objects accept both standard HTTP methods: GET and POST. In our case we only need to GET some data from the server. If we would have to upload (write) data on a server we should use POST instead.


Xcode File Edit View Navigate Editor Product Window Help
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Build Succeeded | Yesterday at 19:23 PM
Nearby Deals
Prototype Cells
Table View
Prototype Content
- (NSArray *)nearbyDeals
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit
NSString *urlString = [NSString stringWithFormat:@"%@?app_key=%@&latitude=%f&longitude=%f&limit=%d",
NSURL *url = [NSURL URLWithString:urlString];
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
timeoutInterval:60];
[request setHTTPMethod:@"GET"];
NSURLConnection *serverConnection = [[NSURLConnection alloc] initWithRequest:request
delegate:self];
return NO;
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
// Return the number of sections.
return 1;
Next, we create the connection with the request and start loading the data. In order to download the contents of a URL, an application needs to provide a delegate object. The download starts immediately upon receiving the initWithRequest:delegate: message.
Our delegate (this Controller) will have a chance to know about the state of the connection by implementing the following delegate methods: connection:didReceiveResponse:, connection:didReceiveData:, connection:didFailWithError: and connectionDidFinishLoading:.


Nearby Deals
Prototype Cells
Table View
Prototype Content
- (NSArray *)nearbyDeals
{
if (_nearbyDeals == nil)
{
_nearbyDeals = [NSArray arrayWithObjects:
@"First Deal",
@"Second Deal",
@"Third Deal",
@"Another Deal",
nil];
}
return _nearbyDeals;
}
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit
{
if (self.webData != nil) return NO;
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
timeoutInterval:60];
[request setHTTPMethod:@"GET"];
NSURLConnection *serverConnection = [[NSURLConnection alloc] initWithRequest:request
delegate:self];
if (serverConnection != nil)
{
self.webData = [NSMutableData data];
return YES;
}
return NO;
}
#pragma mark - Table view data source
- (NSInteger)numberOfSections
{
// Return the number of sections.
return 1;
Create the NSMutableData to hold the received data. This creates an empty data object.
Return YES since we have successfully opened a connection to the server.

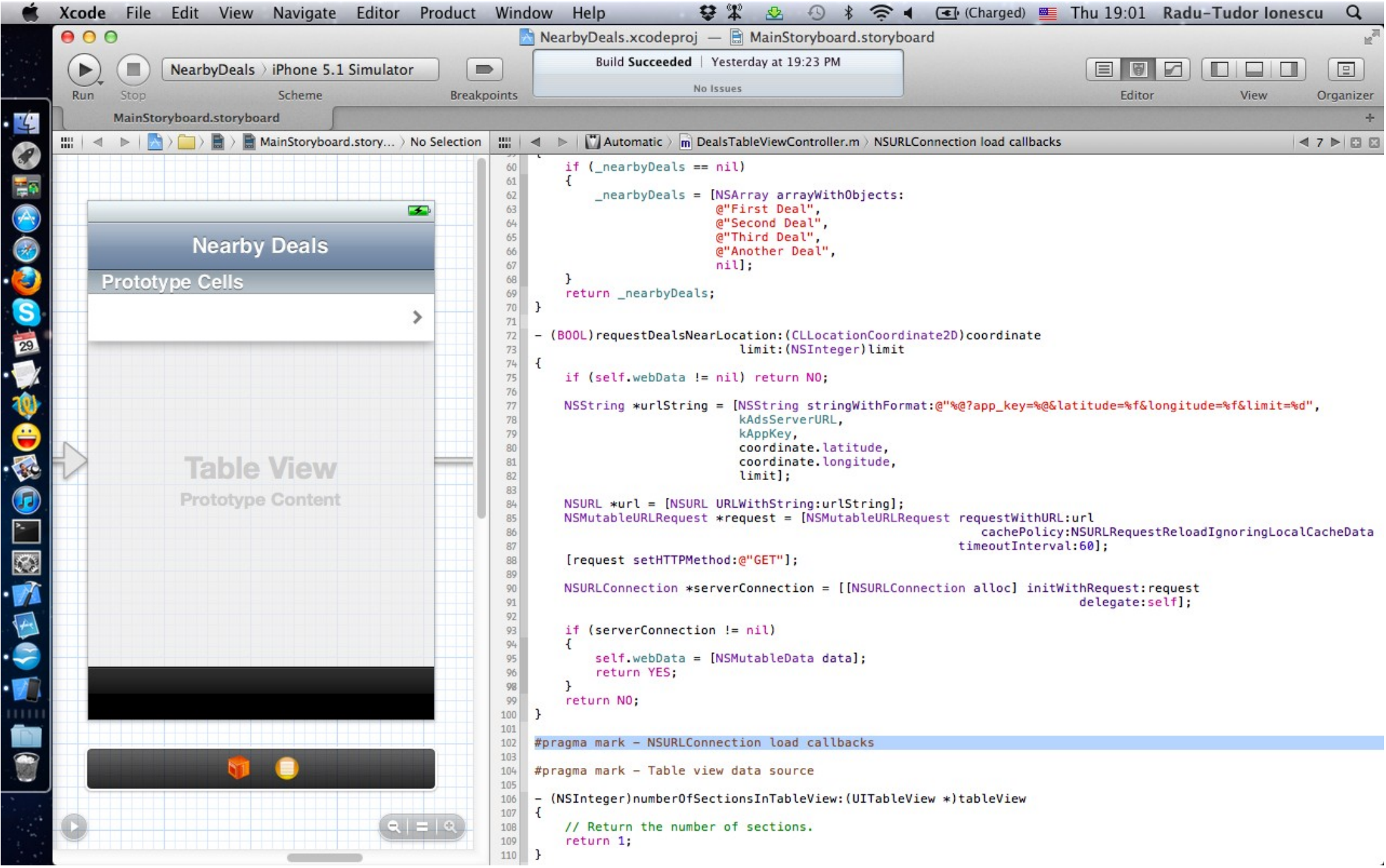
Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

15. We should continue by implementing the methods for the `NSURLConnection` delegate. Let's mark the section of code that will contain these delegate methods using the `#pragma mark` compiler directive.

16. Name this section “NSURLConnection load callbacks”.

The next screenshot shows where to add this section.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 19:01 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.story... No Selection
Nearby Deals
Prototype Cells
Table View
Prototype Content
Automatic DealsTableViewController.m NSURLConnection load callbacks
60 if (_nearbyDeals == nil)
61 {
62 _nearbyDeals = [NSArray arrayWithObjects:
63 @"First Deal",
64 @"Second Deal",
65 @"Third Deal",
66 @"Another Deal",
67 nil];
68 }
69 return _nearbyDeals;
70 }
71
72 - (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
73 limit:(NSInteger)limit
74 {
75 if (self.webData != nil) return NO;
76
77 NSString *urlString = [NSString stringWithFormat:@"%@?app_key=%@&latitude=%f&longitude=%f&limit=%d",
78 kAdsServerURL,
79 kAppKey,
80 coordinate.latitude,
81 coordinate.longitude,
82 limit];
83
84 NSURL *url = [NSURL URLWithString:urlString];
85 NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
86 cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
87 timeoutInterval:60];
88 [request setHTTPMethod:@"GET"];
89
90 NSURLConnection *serverConnection = [[NSURLConnection alloc] initWithRequest:request
91 delegate:self];
92
93 if (serverConnection != nil)
94 {
95 self.webData = [NSMutableData data];
96 return YES;
98 }
99 return NO;
100 }
101
102 #pragma mark - NSURLConnection load callbacks
103
104 #pragma mark - Table view data source
105
106 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
107 {
108 // Return the number of sections.
109 return 1;
110 }

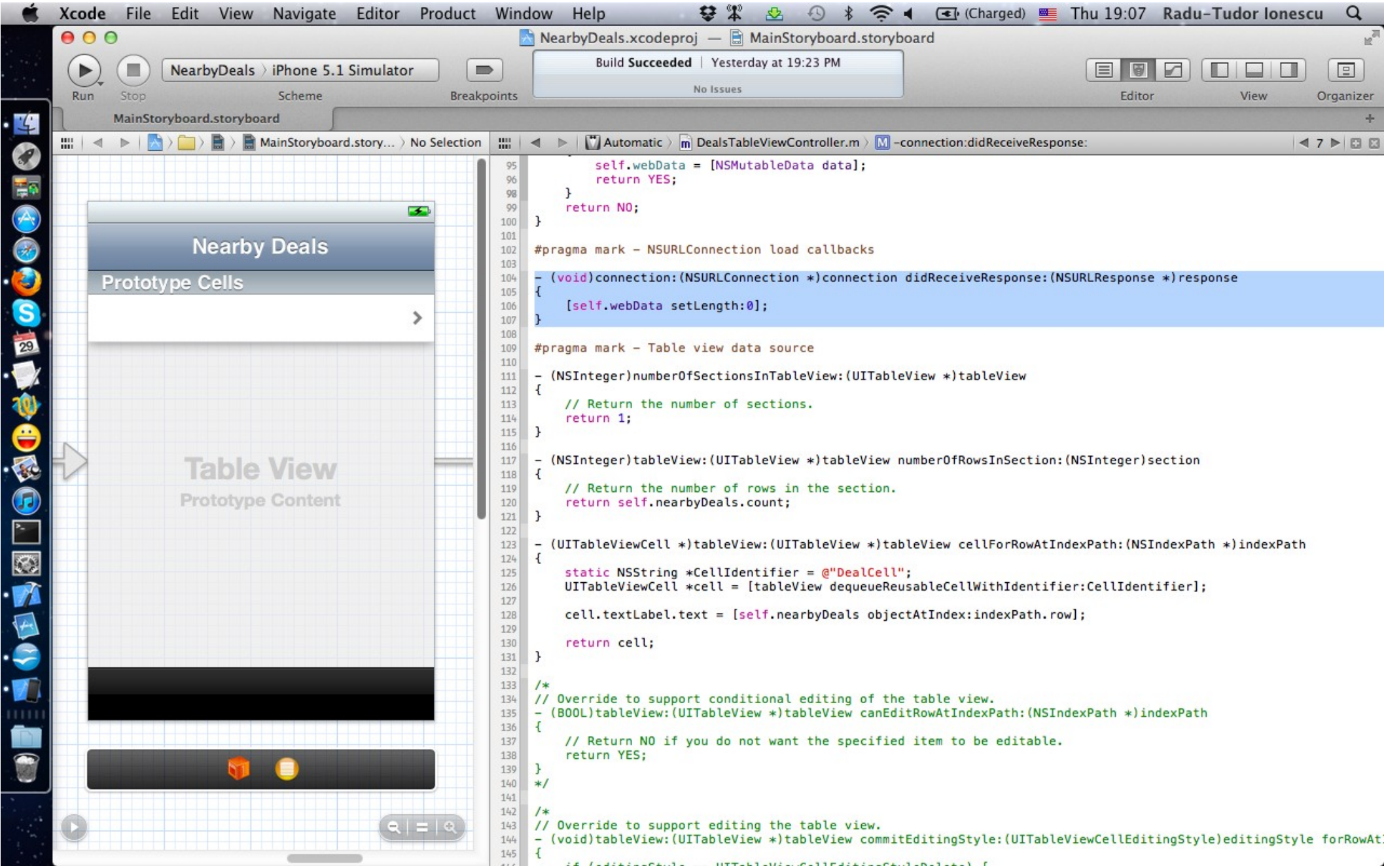
Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

17. When the server has provided sufficient data to create an `NSURLResponse` object, the `delegate` receives the following message: `connection:didReceiveResponse:`.

You should be prepared for your `delegate` to receive the `connection:didReceiveResponse:` message multiple times for a single connection. This message can be sent due to server redirects, or in rare cases multi-part MIME documents. Each time the delegate receives the `connection:didReceiveResponse:` message, it should reset any progress indication and discard all previously received data.

The next screenshot shows how to implement this callback method.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 19:07 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.story... No Selection
Automatic DealsTableViewController.m -connection:didReceiveResponse:
Nearby Deals
Prototype Cells
Table View
Prototype Content
95 self.webData = [NSMutableData data];
96 return YES;
98 }
99 return NO;
100 }
101
102 #pragma mark - NSURLConnection load callbacks
103
104 - (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
105 {
106 [self.webData setLength:0];
107 }
108
109 #pragma mark - Table view data source
110
111 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
112 {
113 // Return the number of sections.
114 return 1;
115 }
116
117 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
118 {
119 // Return the number of rows in the section.
120 return self.nearbyDeals.count;
121 }
122
123 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
124 {
125 static NSString *CellIdentifier = @"DealCell";
126 UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
127
128 cell.textLabel.text = [self.nearbyDeals objectAtIndex:indexPath.row];
129
130 return cell;
131 }
132
133 /*
134 // Override to support conditional editing of the table view.
135 - (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
136 {
137 // Return NO if you do not want the specified item to be editable.
138 return YES;
139 }
140 */
141
142 /*
143 // Override to support editing the table view.
144 - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAt
145 {

Task 2

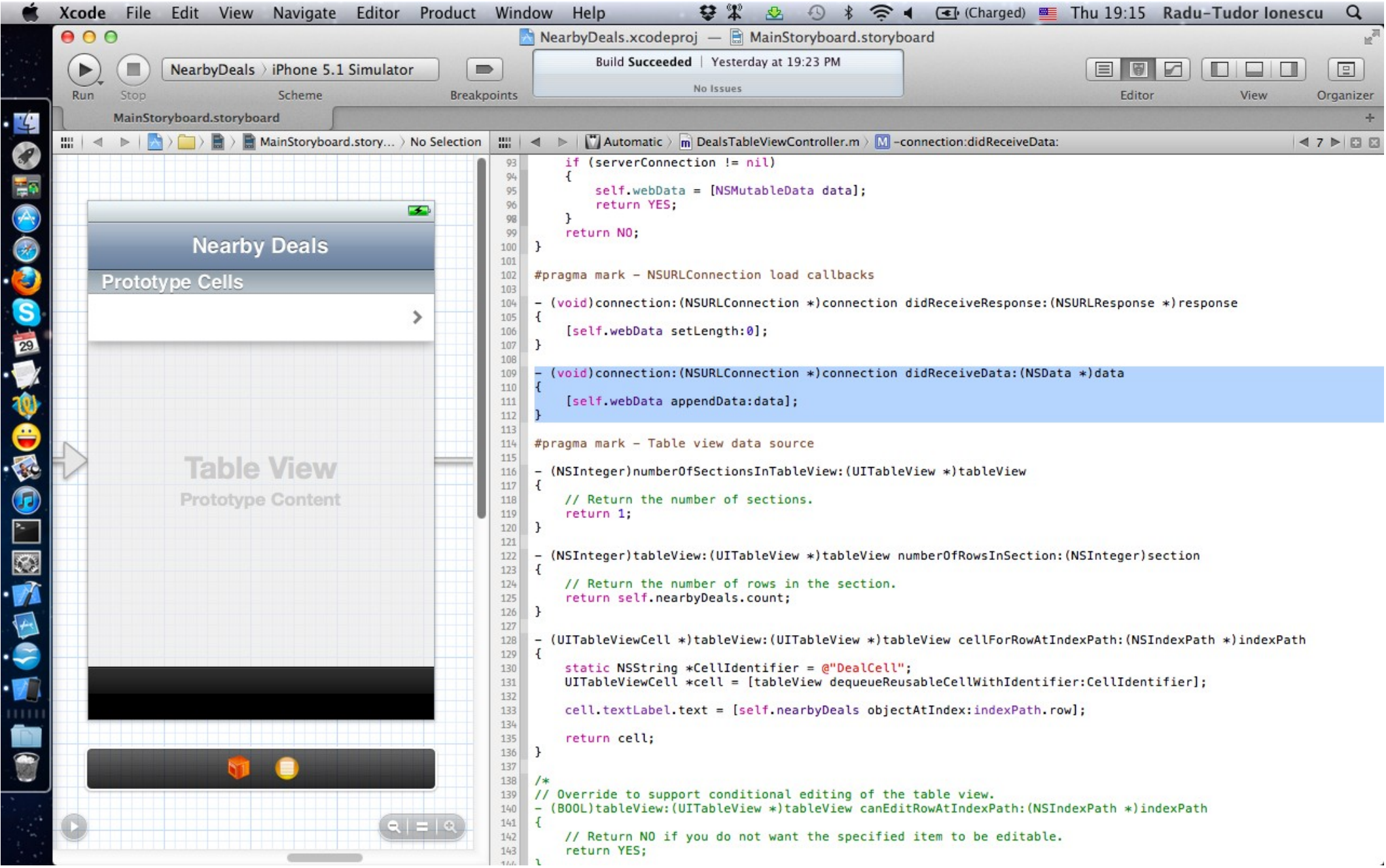
Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

18. Our Table View Controller (the `delegate`) is periodically sent `connection:didReceiveData:` messages as the data is received. The delegate implementation is responsible for storing the newly received data.

We should append the new data to `webData` in this method.

Note that you can also use this method to provide an indication of the connection's progress to the user. This is useful when we transfer large files from/to a server.

The next screenshot shows how to implement this callback method.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 19:15 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.story... No Selection
Automatic DealsTableViewController.m -connection:didReceiveData:
Nearby Deals
Prototype Cells
Table View
Prototype Content
93 if (serverConnection != nil)
94 {
95 self.webData = [NSMutableData data];
96 return YES;
98 }
99 return NO;
100 }
101
102 #pragma mark - NSURLConnection load callbacks
103
104 - (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
105 {
106 [self.webData setLength:0];
107 }
108
109 - (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
110 {
111 [self.webData appendData:data];
112 }
113
114 #pragma mark - Table view data source
115
116 - (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
117 {
118 // Return the number of sections.
119 return 1;
120 }
121
122 - (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
123 {
124 // Return the number of rows in the section.
125 return self.nearbyDeals.count;
126 }
127
128 - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
129 {
130 static NSString *CellIdentifier = @"DealCell";
131 UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
132
133 cell.textLabel.text = [self.nearbyDeals objectAtIndex:indexPath.row];
134
135 return cell;
136 }
137
138 /*
139 // Override to support conditional editing of the table view.
140 - (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
141 {
142 // Return NO if you do not want the specified item to be editable.
143 return YES;

Task 2

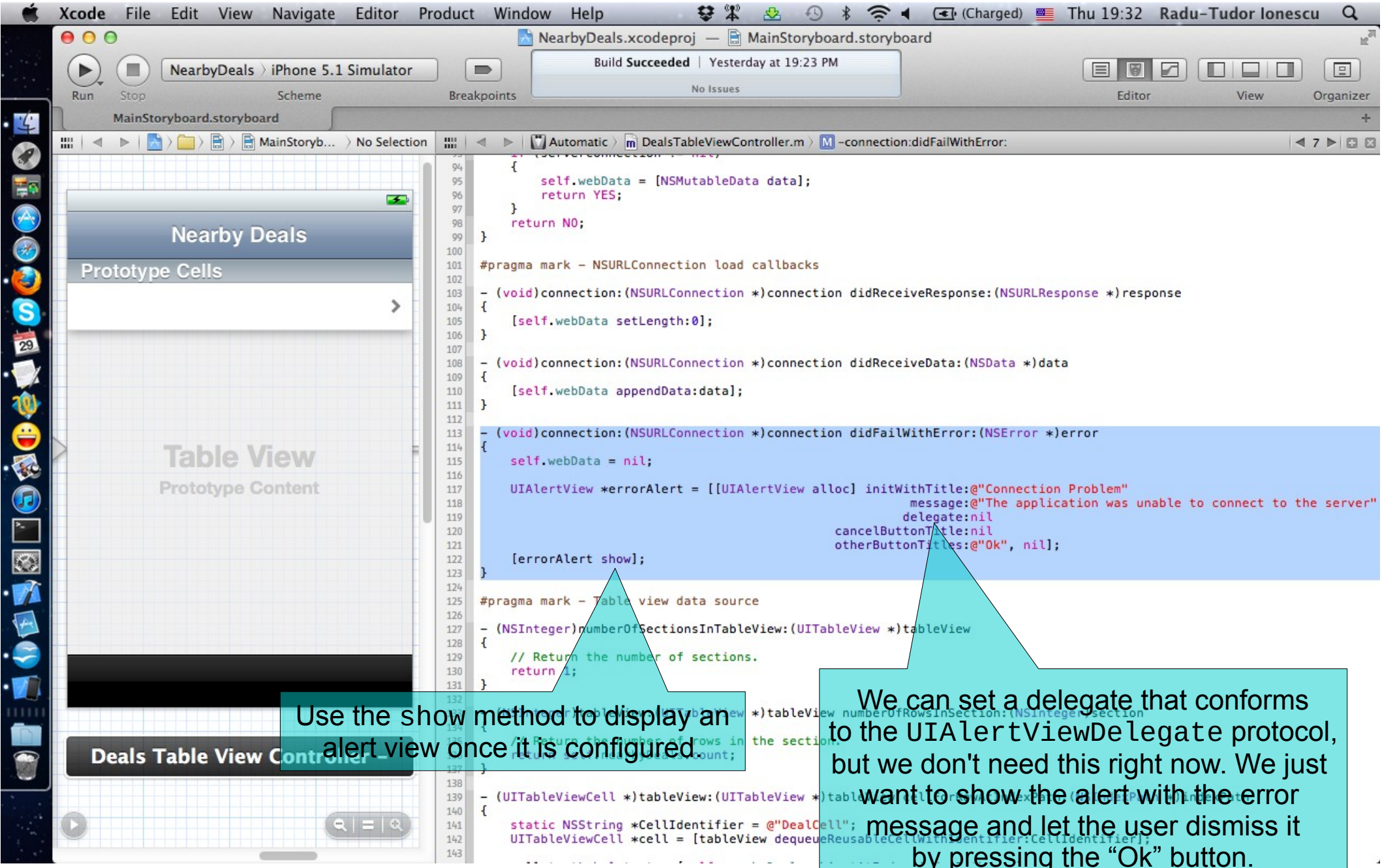
Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

19. If an error is encountered during the download, the delegate receives a `connection:didFailWithError:` message. The `NSError` object passed as the parameter specifies the details of the error. After our delegate object receives the `connection:didFailWithError:` message, it receives no further delegate messages for the specified connection.

We set `webData` to `nil` in this method because we have finished the request (without success).

20. It would be nice to let the user know that we have a connection problem. A standard way to present ad-hoc messages to the user is to use an `UIAlertView` object. We are going to build an alert programmatically that will display an error message on the screen. It will have a dismiss button too.

The next screenshot shows how to implement this callback method.


Xcode File Edit View Navigate Editor Product Window Help
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Build Succeeded | Yesterday at 19:23 PM
No Issues
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller
Automatic DealsTableViewController.m -connection:didFailWithError:
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
{
self.webData = nil;
UIAlertView *errorAlert = [[UIAlertView alloc] initWithTitle:@"Connection Problem"
message:@"The application was unable to connect to the server"
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"Ok", nil];
[errorAlert show];
}
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
// Return the number of sections.
return 1;
}
Use the show method to display an alert view once it is configured.
We can set a delegate that conforms to the UIAlertViewDelegate protocol, but we don't need this right now. We just want to show the alert with the error message and let the user dismiss it by pressing the “Ok” button.

Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

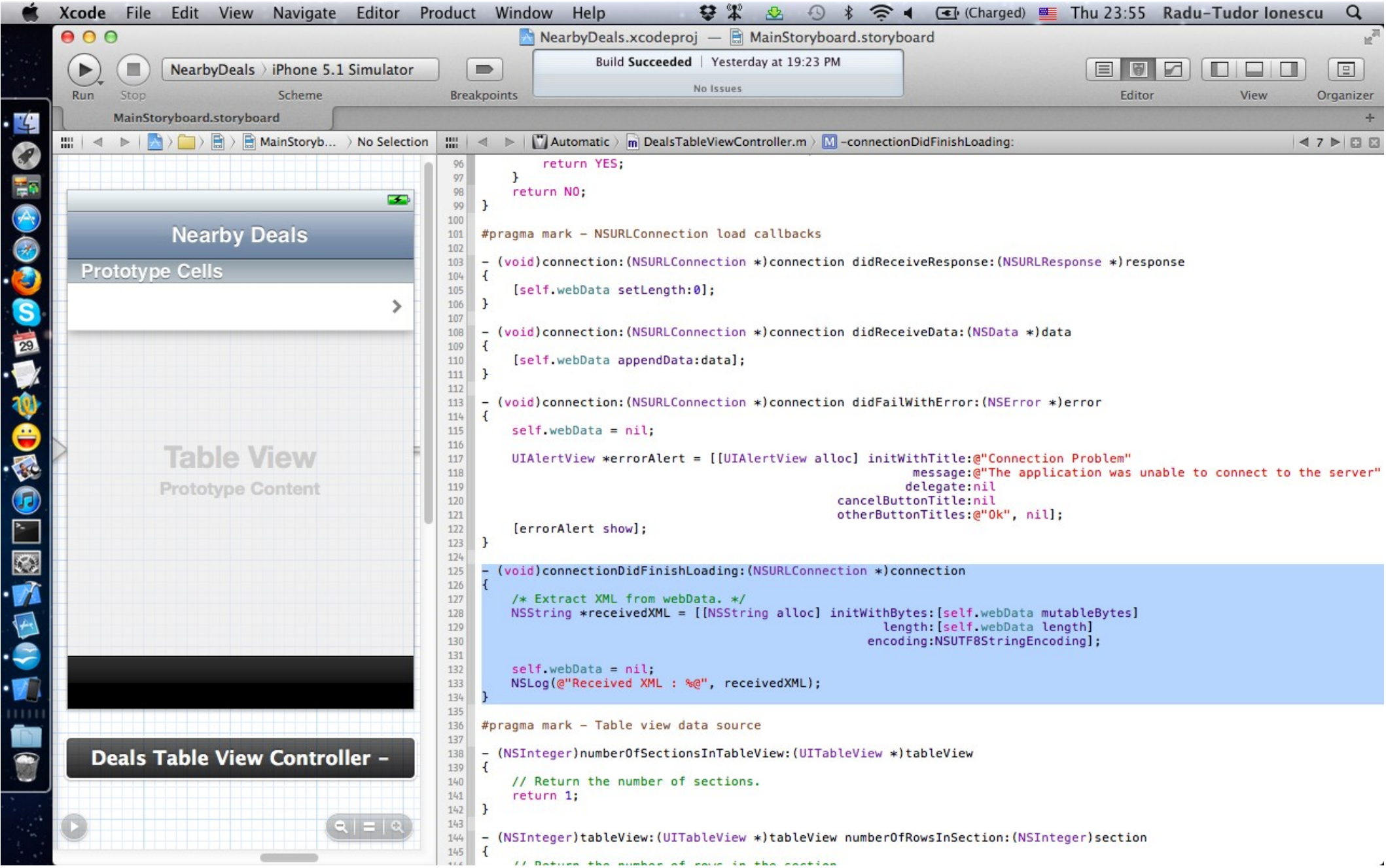
21. Finally, if the connection succeeds in downloading the request, the delegate receives the `connectionDidFinishLoading:` message. The delegate will receive no further messages for the connection.

Let's transform the received bytes into a string and print it with an `NSLog` for now.

This represents the simplest implementation of a client using `NSURLConnection`. Additional delegate methods provide the ability to customize the handling of server redirects, authorization requests and caching of the response.

The next screenshot shows how to implement this callback method.

22. Run the application in iOS Simulator and check out the console to see if it gets the XML from the server.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Thu 23:55 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Yesterday at 19:23 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryb... No Selection
Automatic DealsTableViewController.m -connectionDidFinishLoading:
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
return YES;
}
return NO;
}
#pragma mark - NSURLConnection load callbacks
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
[self.webData setLength:0];
}
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
[self.webData appendData:data];
}
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
{
self.webData = nil;
UIAlertView *errorAlert = [[UIAlertView alloc] initWithTitle:@"Connection Problem"
message:@"The application was unable to connect to the server"
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"Ok", nil];
[errorAlert show];
}
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
/* Extract XML from webData. */
NSString *receivedXML = [[NSString alloc] initWithBytes:[self.webData mutableBytes]
length:[self.webData length]
encoding:NSUTF8StringEncoding];
self.webData = nil;
NSLog(@"Received XML : %@", receivedXML);
}
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
// Return the number of sections.
return 1;
}
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{

Task 2

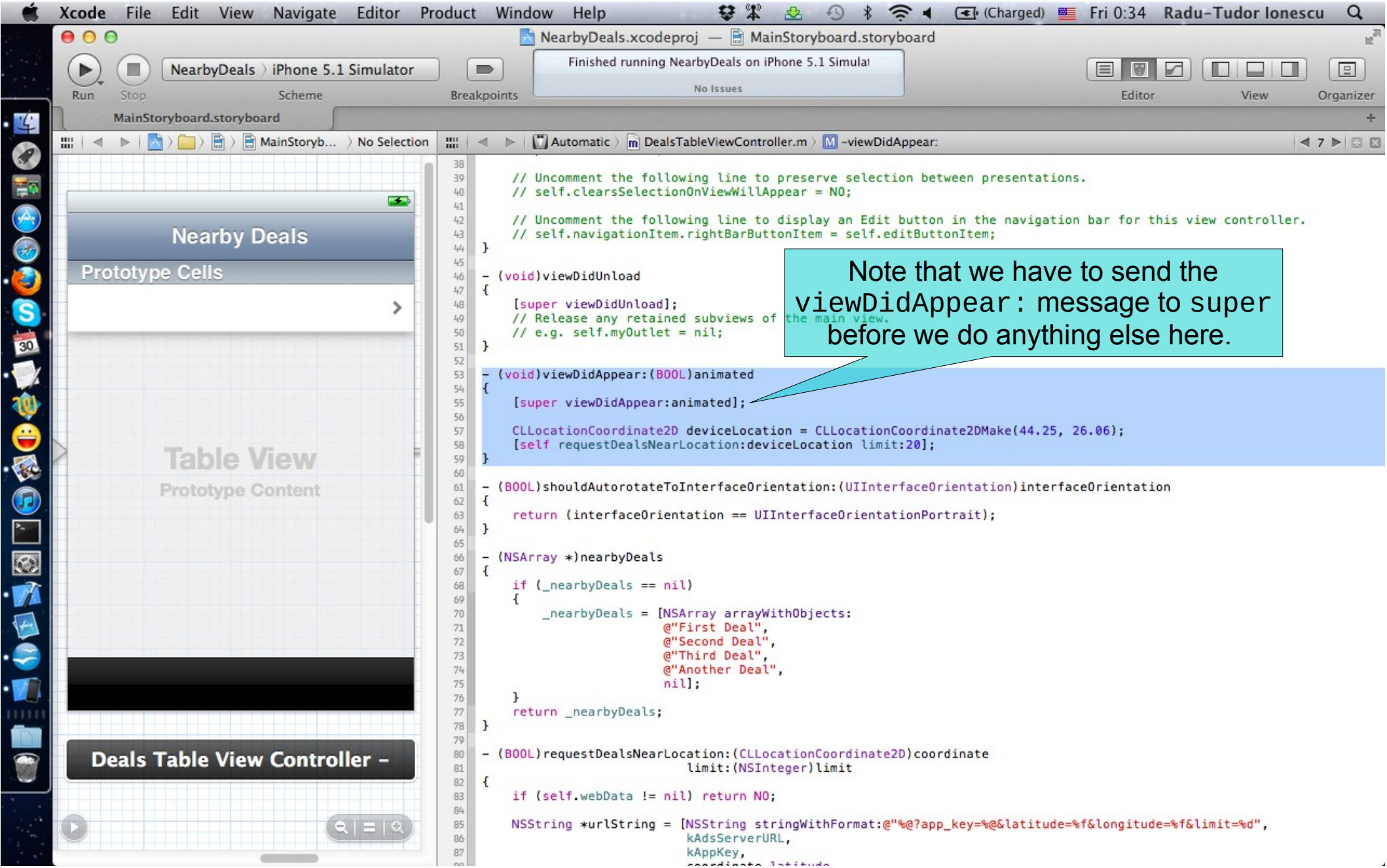
Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants.

22. We want to make this request right when our Table View appears on screen. Note the each View Controller has a lifecycle that starts with its creation. We will discuss this later in detail, but for now you should know that you (the Controller) get notified when your View appears on screen.

If we want to do something custom (such as making a request to a server) when it appears on screen we should implement the `viewDidAppear:` method.

We are going to make a request from a fake location that we create using the `CLLocationCoordinate2DMake` helper C function. Later we will use the GPS location of the device.

The next screenshot shows how to implement this callback method.


Xcode File Edit View Navigate Editor Product Window Help
Fri 0:34 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Finished running NearbyDeals on iPhone 5.1 Simula
No Issues
Editor View Organizer
MainStoryboard.storyboard
MainStoryb... No Selection
Automatic DealsTableViewController.m -viewDidAppear:
Nearby Deals
Prototype Cells
Table View
Prototype Content
Deals Table View Controller –
Note that we have to send the viewDidAppear: message to super before we do anything else here.
// Uncomment the following line to preserve selection between presentations.
// self.clearsSelectionOnViewWillAppear = NO;
// Uncomment the following line to display an Edit button in the navigation bar for this view controller.
// self.navigationItem.rightBarButtonItem = self.editButtonItem;
}
- (void)viewDidUnload
{
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (void)viewDidAppear:(BOOL)animated
{
[super viewDidAppear:animated];
CLLocationCoordinate2D deviceLocation = CLLocationCoordinate2DMake(44.25, 26.06);
[self requestDealsNearLocation:deviceLocation limit:20];
}
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
- (NSArray *)nearbyDeals
{
if (_nearbyDeals == nil)
{
_nearbyDeals = [NSArray arrayWithObjects:
@"First Deal",
@"Second Deal",
@"Third Deal",
@"Another Deal",
nil];
}
return _nearbyDeals;
}
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit
{
if (self.webData != nil) return NO;
NSString *urlString = [NSString stringWithFormat:@"%@?app_key=%@&latitude=%f&longitude=%f&limit=%d",
kAdsServerURL,
kAppKey,

Task 2

Task: Create a request using the GeoAds+ API that will return deals from nearby Bars or Restaurants..

23. Run the application in iOS Simulator and check out the console to see if it gets the XML from the server.

24. Stop running the application.

25. Disconnect you computer from the Internet (just pull out the cable) and lets see what happens with our request.

Run the application again. This time it should display the error message.

26. Stop running the application.

27. Don't forget to put back your Internet cable.

Task 3

Task: Add a new class to your project with helper methods for XML parsing.

1. Usually we use the `NSXMLParser` class to parse XML documents. This class parses XML files in an event-driven manner as it notifies its delegate about the items (elements, attributes) that it encounters as it processes an XML document. This is very efficient when we want to parse large XML files, but in our case we need something simple. Thus, we are going to parse the XML file ourself and build an `NSArray` with the nearby deals.

Actually, we are going to add SimpleXMLParser.h/m to our project. This class already contains some helper class methods for parsing XML documents.

Open Project Navigator.

2. Copy and paste SimpleXMLParser.h and SimpleXMLParser.m to the NearbyDeals folder within your project folder. Remember that you can right-click on the NearbyDeals group in Project Navigator and select “Show in Finder”.

Task 3

Task: Add a new class to your project with helper methods for XML parsing.

3. Right-click on the NearbyDeals group in Project Navigator and select “Add files to NearbyDeals...”.

4. Select the SimpleXMLParser.h and SimpleXMLParser.m from the NearbyDeals folder and click Add.

5. We should add some functionality to the `SimpleXMLParser` class in order to parse the XML with nearby deals. It's good if we do this in another tab. Go to “File > New > Tab” in Xcode menu or use the CMD+T shortcut keys to create the new tab.

6. Open SimpleXMLParser.m on the left-side of the Editor, and SimpleXMLParser.h on the right-side. The header file should automatically appear on the right-side if you are on automatic mode.

7. Take a look at the implementation of the following methods:
   `convertSpecialCharactersToUnicodeInXML:`
   `contentOfFirstTagWithName:fromXML:`

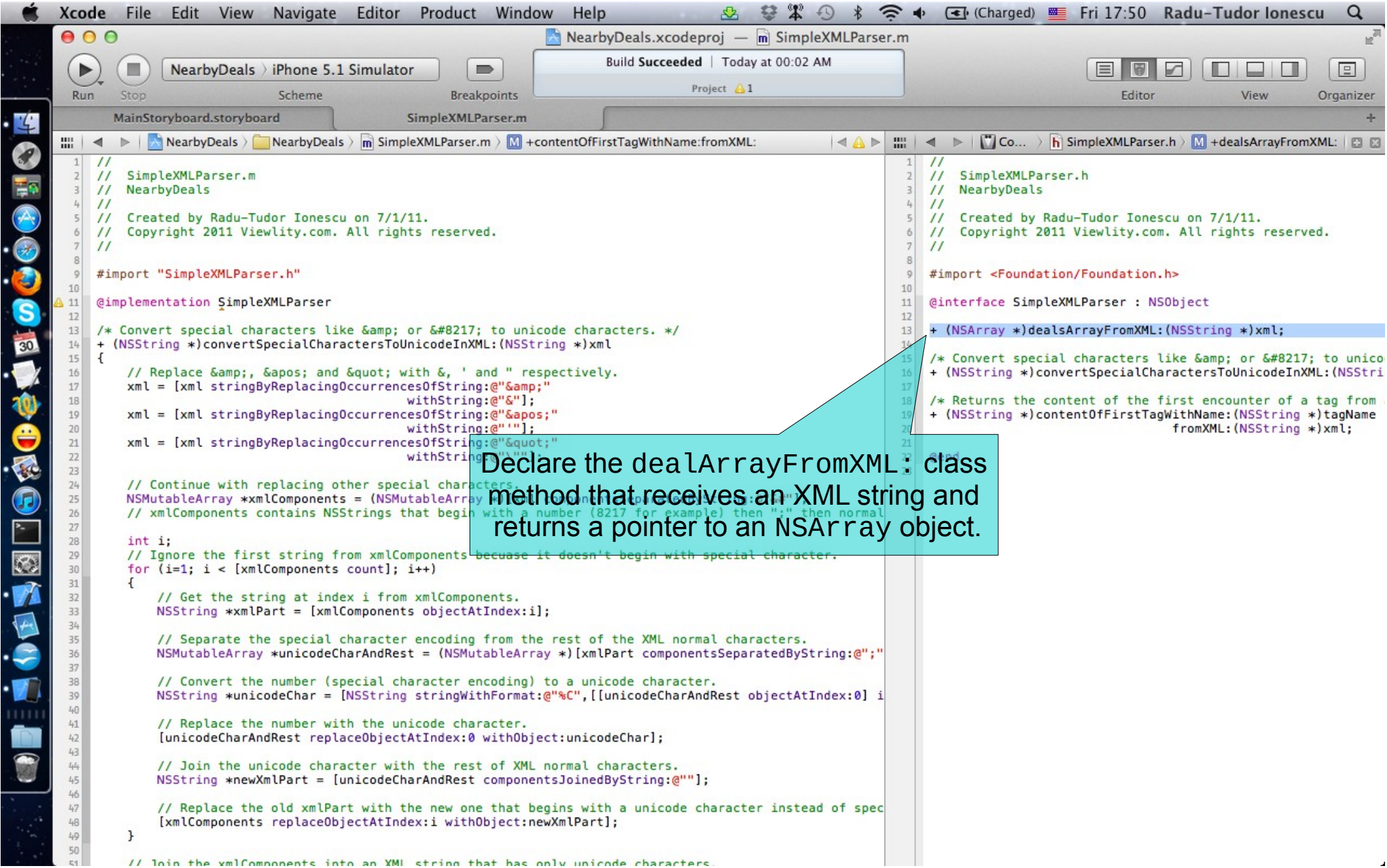
Task 3

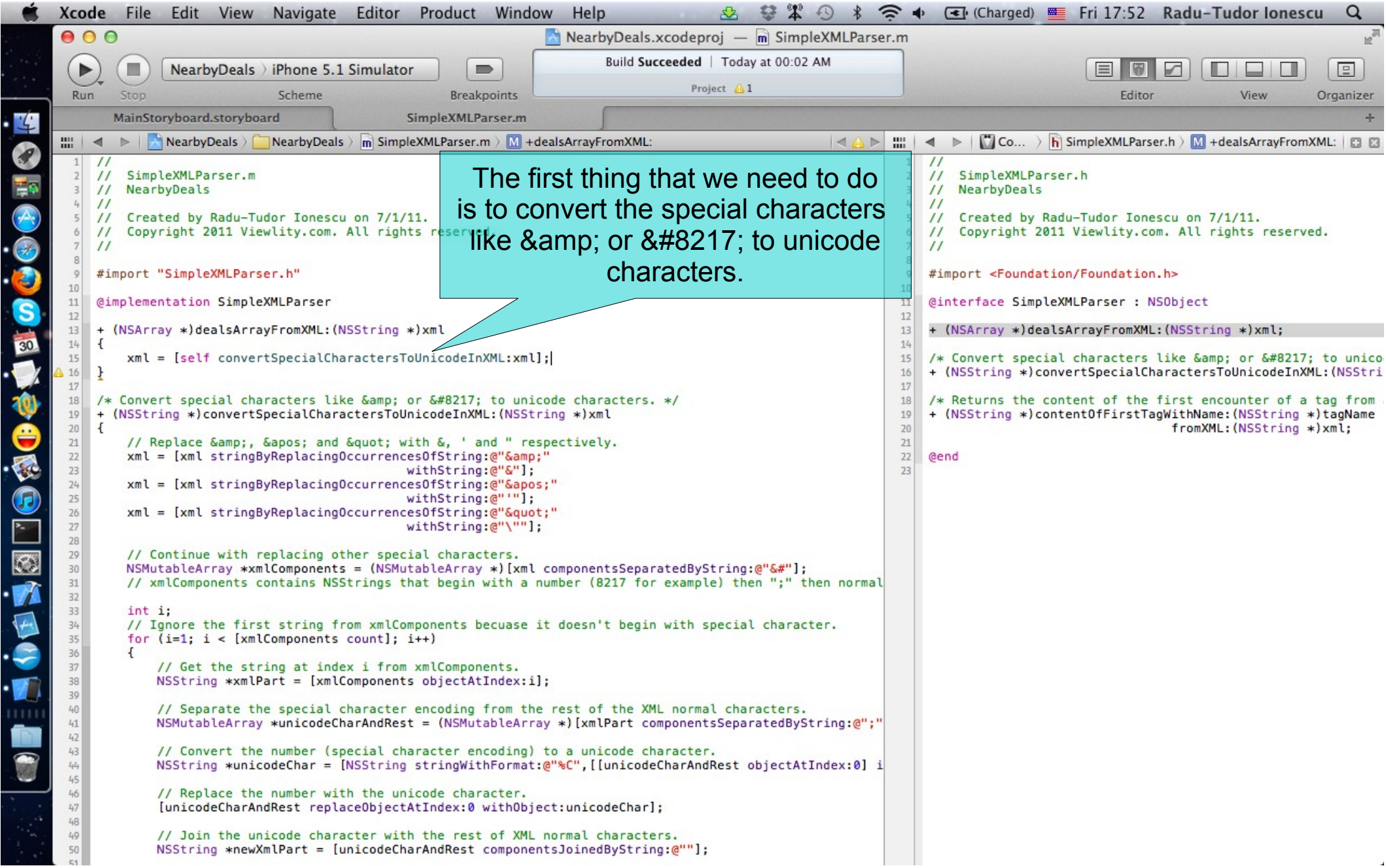
Task: Add a new class to your project with helper methods for XML parsing.

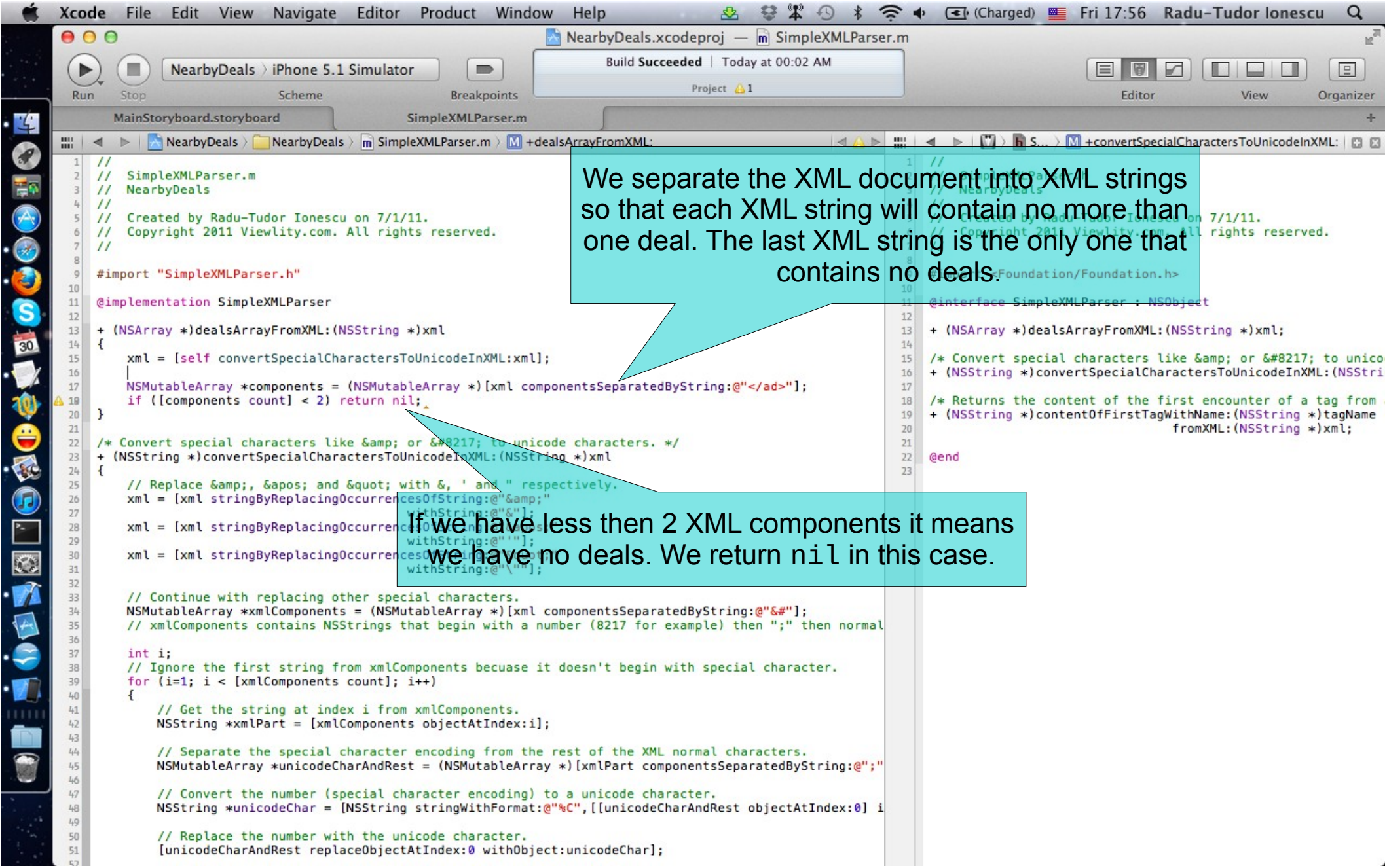
8. Let's declare and implement another class method in the `SimpleXMLParser` that will parse the XML document with nearby deals received from the server.

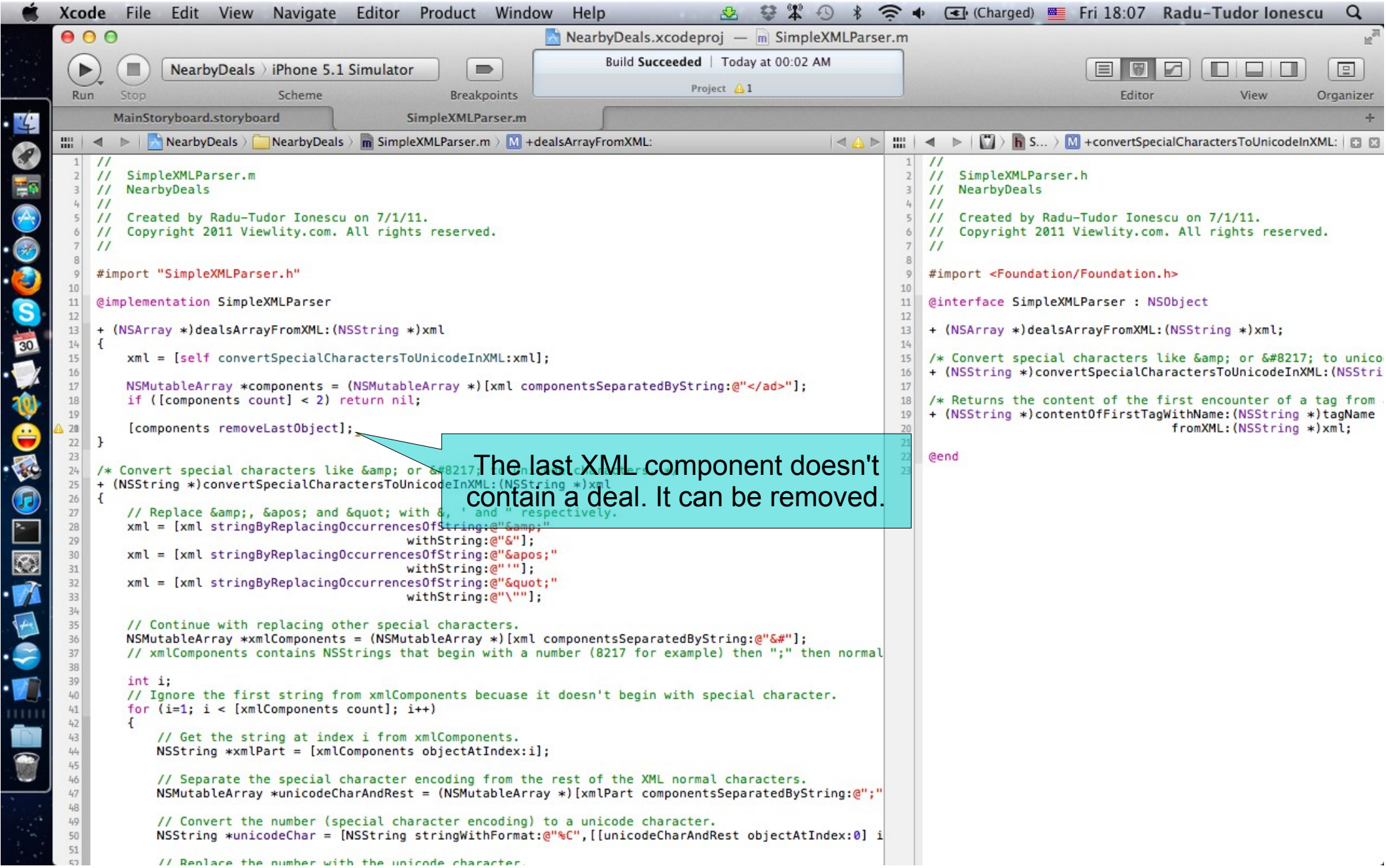
It will build an array of deals. Each deal will be represented by an `NSDictionary` that will store the deal's title, subtitle, description, etc.

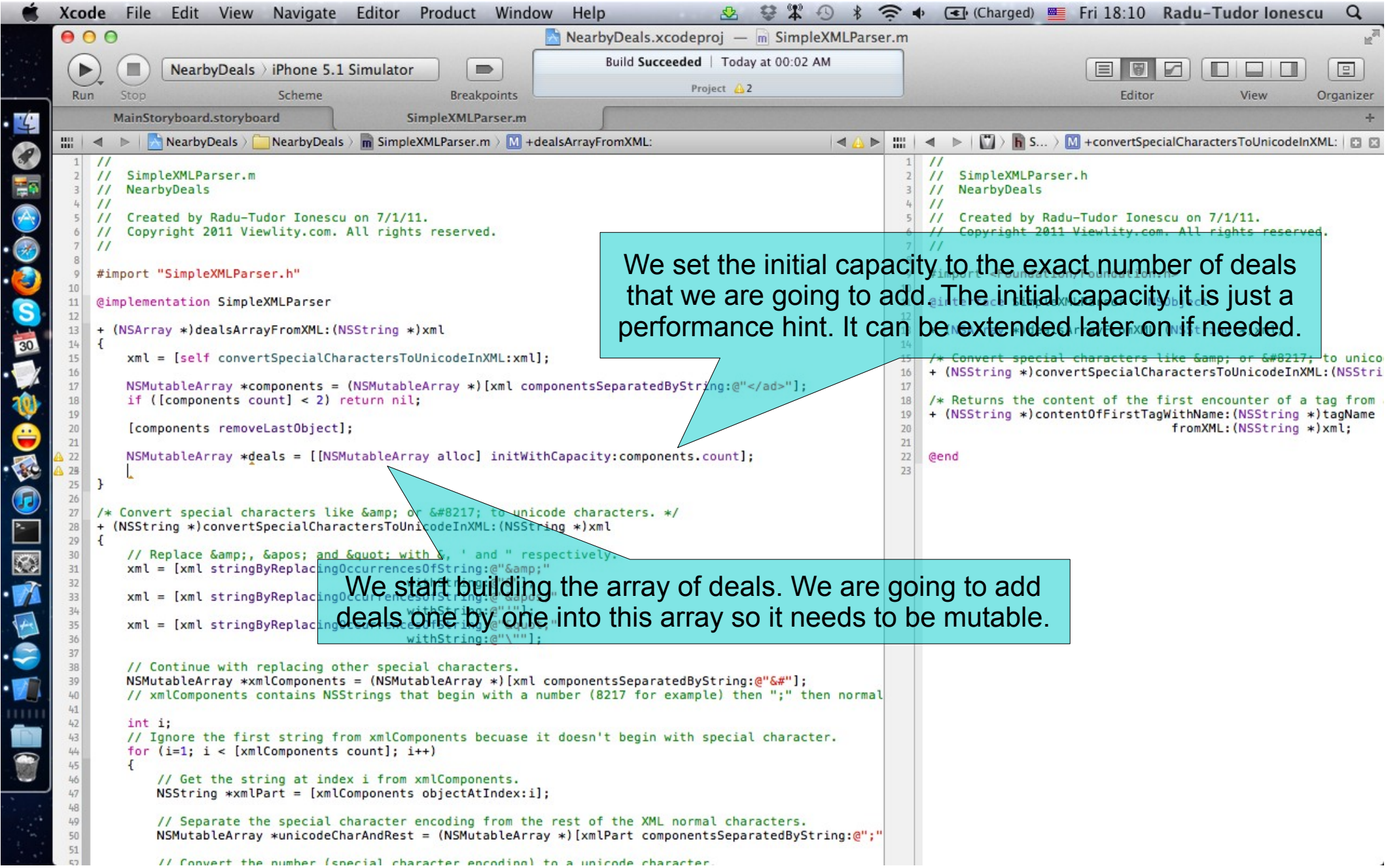
Follow the steps from the next slides to implement this method.

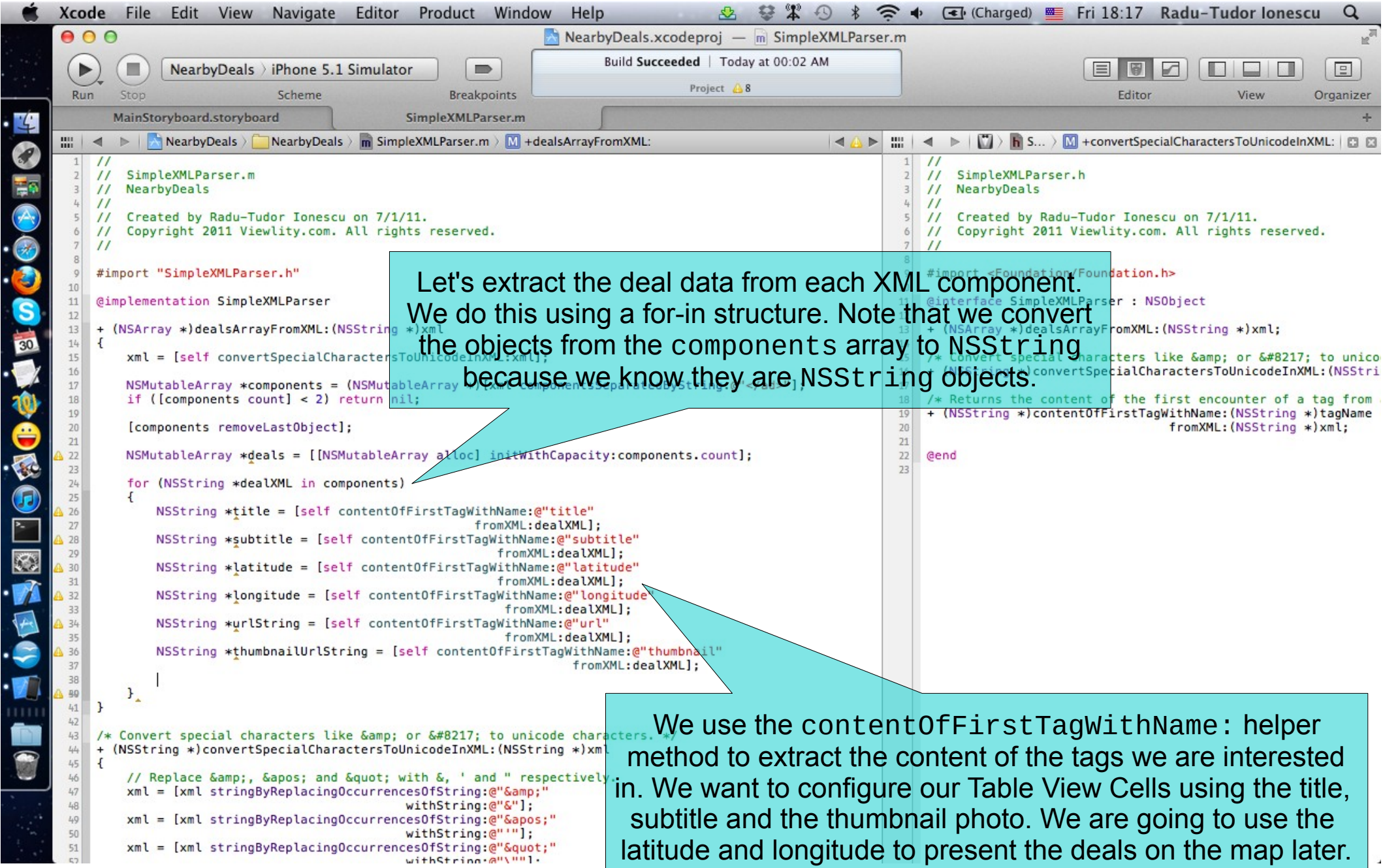

Declare the dealArrayFromXML: class method that receives an XML string and returns a pointer to an NSArray object.

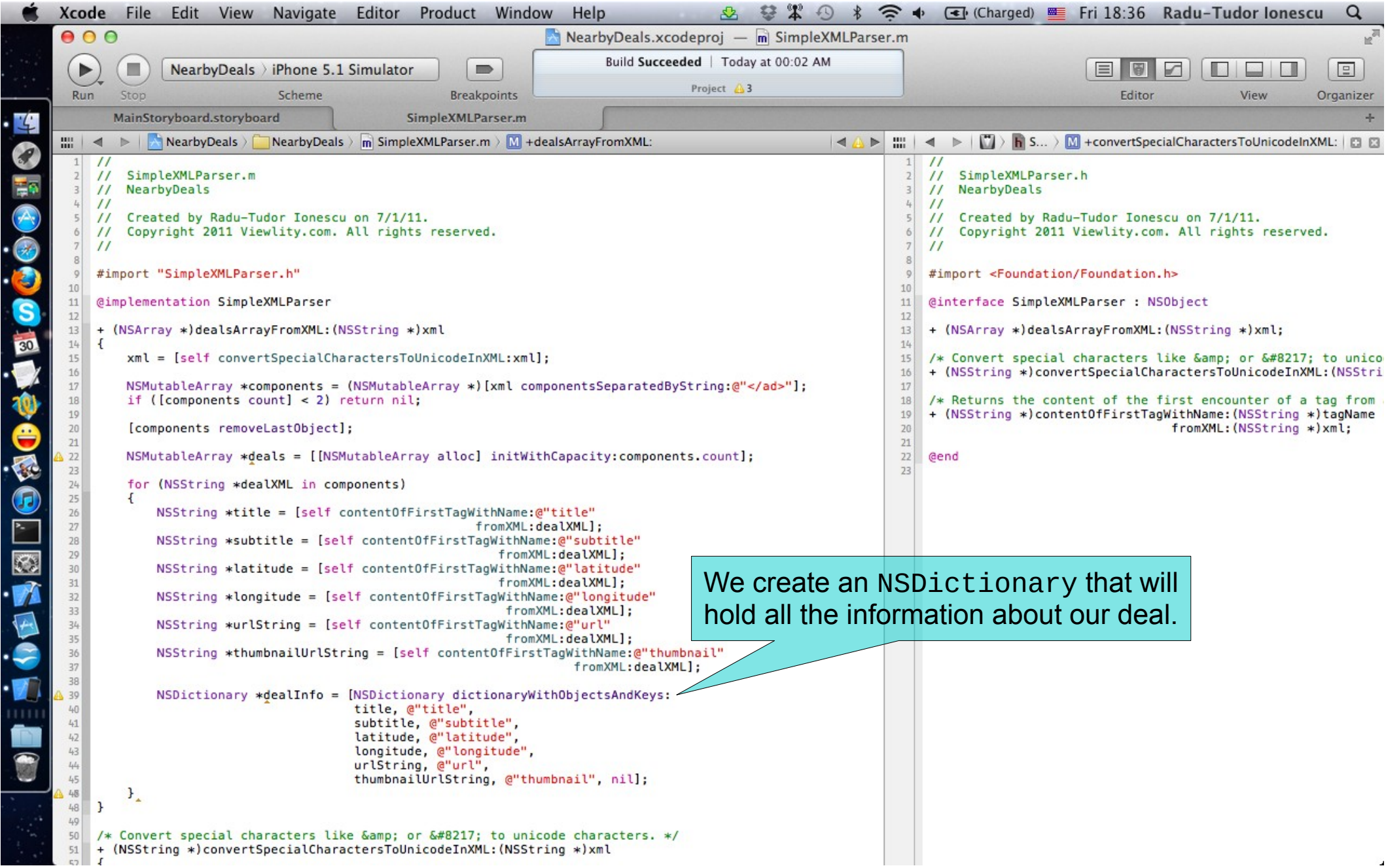

Xcode File Edit View Navigate Editor Product Window Help
(Charged) Fri 17:52 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
Run Stop NearbyDeals › iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Today at 00:02 AM
Project 1
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals › NearbyDeals › SimpleXMLParser.m › +dealsArrayFromXML:
Co... › SimpleXMLParser.h › +dealsArrayFromXML:
The first thing that we need to do is to convert the special characters like & or ’ to unicode characters.
//
// SimpleXMLParser.m
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 7/1/11.
// Copyright 2011 Viewlity.com. All rights reserved.
//
#import "SimpleXMLParser.h"
@implementation SimpleXMLParser
+ (NSArray *)dealsArrayFromXML:(NSString *)xml
{
xml = [self convertSpecialCharactersToUnicodeInXML:xml];
}
/* Convert special characters like & or ’ to unicode characters. */
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSString *)xml
{
// Replace &, ' and " with &, ' and " respectively.
xml = [xml stringByReplacingOccurrencesOfString:@"&"
withString:@"&"];
xml = [xml stringByReplacingOccurrencesOfString:@"'"
withString:@"'"];
xml = [xml stringByReplacingOccurrencesOfString:@"""
withString:@"\""];
// Continue with replacing other special characters.
NSMutableArray *xmlComponents = (NSMutableArray *)[xml componentsSeparatedByString:@"&#"];
// xmlComponents contains NSStrings that begin with a number (8217 for example) then ";" then normal
int i;
// Ignore the first string from xmlComponents becuase it doesn't begin with special character.
for (i=1; i < [xmlComponents count]; i++)
{
// Get the string at index i from xmlComponents.
NSString *xmlPart = [xmlComponents objectAtIndex:i];
// Separate the special character encoding from the rest of the XML normal characters.
NSMutableArray *unicodeCharAndRest = (NSMutableArray *)[xmlPart componentsSeparatedByString:@";"
// Convert the number (special character encoding) to a unicode character.
NSString *unicodeChar = [NSString stringWithFormat:@"%C",[[unicodeCharAndRest objectAtIndex:0] i
// Replace the number with the unicode character.
[unicodeCharAndRest replaceObjectAtIndex:0 withObject:unicodeChar];
// Join the unicode character with the rest of XML normal characters.
NSString *newXmlPart = [unicodeCharAndRest componentsJoinedByString:@""];
//
// SimpleXMLParser.h
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 7/1/11.
// Copyright 2011 Viewlity.com. All rights reserved.
//
#import <Foundation/Foundation.h>
@interface SimpleXMLParser : NSObject
+ (NSArray *)dealsArrayFromXML:(NSString *)xml;
/* Convert special characters like & or ’ to unico
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSStri
/* Returns the content of the first encounter of a tag from
+ (NSString *)contentOfFirstTagWithName:(NSString *)tagName
fromXML:(NSString *)xml;
@end

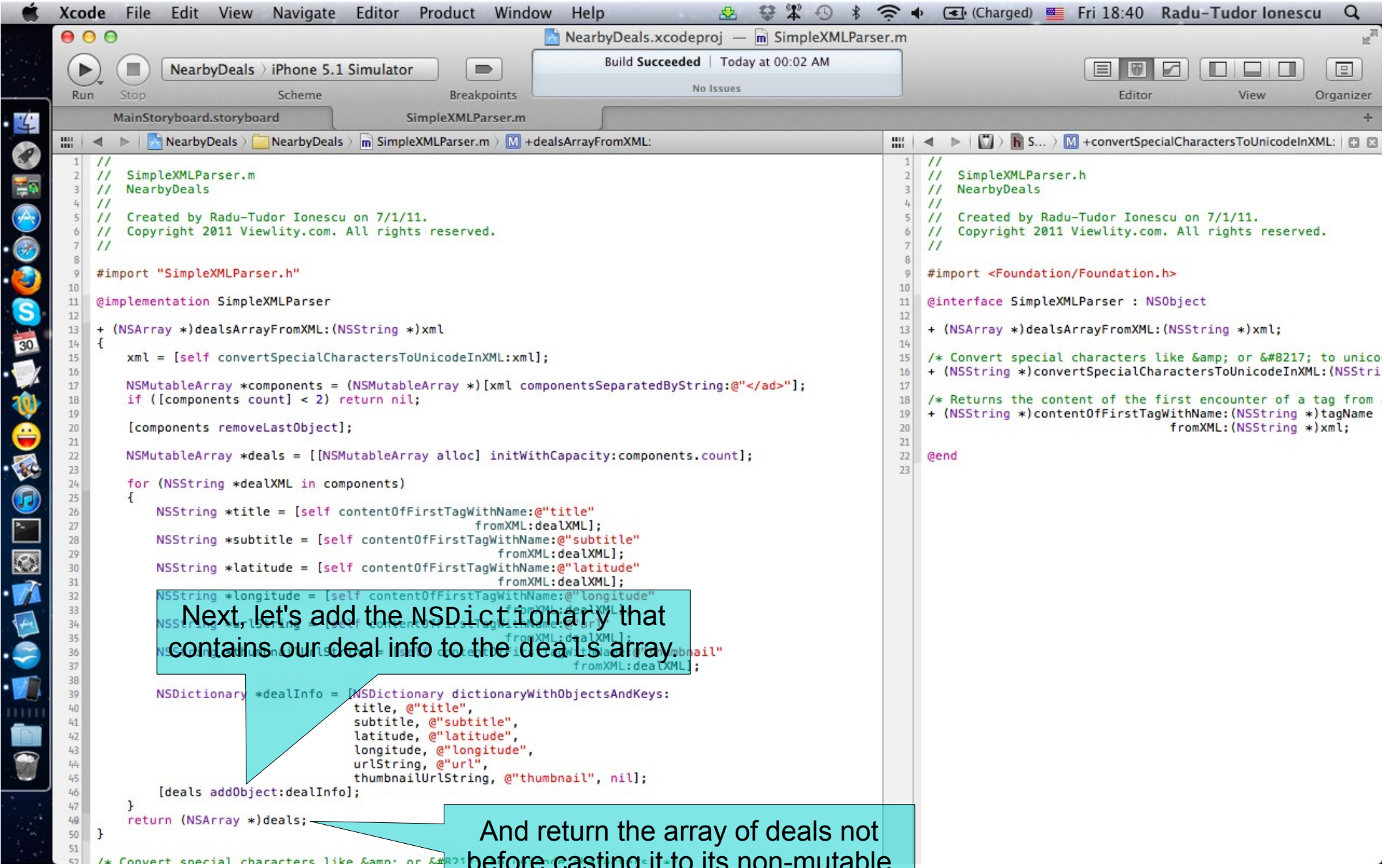

We separate the XML document into XML strings so that each XML string will contain no more than one deal. The last XML string is the only one that contains no deals.
If we have less then 2 XML components it means we have no deals. We return nil in this case.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Fri 18:07 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
Run Stop NearbyDeals › iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Today at 00:02 AM
Project 1
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals › NearbyDeals › SimpleXMLParser.m › +dealsArrayFromXML:
S... › +convertSpecialCharactersToUnicodeInXML:
The last XML component doesn't contain a deal. It can be removed.


Xcode File Edit View Navigate Editor Product Window Help
Fri 18:10 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
NearbyDeals › iPhone 5.1 Simulator
Build Succeeded | Today at 00:02 AM
Project 2
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals › NearbyDeals › SimpleXMLParser.m › +dealsArrayFromXML:
+convertSpecialCharactersToUnicodeInXML:
We set the initial capacity to the exact number of deals that we are going to add. The initial capacity it is just a performance hint. It can be extended later on if needed.
We start building the array of deals. We are going to add deals one by one into this array so it needs to be mutable.


Xcode File Edit View Navigate Editor Product Window Help
Fri 18:17 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
NearbyDeals › iPhone 5.1 Simulator
Build Succeeded | Today at 00:02 AM
Project 8
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals › NearbyDeals › SimpleXMLParser.m › +dealsArrayFromXML:
S... › +convertSpecialCharactersToUnicodeInXML:
// SimpleXMLParser.m // NearbyDeals // Created by Radu-Tudor Ionescu on 7/1/11. // Copyright 2011 Viewlity.com. All rights reserved. //
#import "SimpleXMLParser.h"
@implementation SimpleXMLParser
+ (NSArray *)dealsArrayFromXML:(NSString *)xml
{
xml = [self convertSpecialCharactersToUnicodeInXML:xml];
NSMutableArray *components = (NSMutableArray *)[xml componentsSeparatedByString:@"</deal>"];
if ([components count] < 2) return nil;
[components removeLastObject];
NSMutableArray *deals = [[NSMutableArray alloc] initWithCapacity:components.count];
for (NSString *dealXML in components)
{
NSString *title = [self contentOfFirstTagWithName:@"title" fromXML:dealXML];
NSString *subtitle = [self contentOfFirstTagWithName:@"subtitle" fromXML:dealXML];
NSString *latitude = [self contentOfFirstTagWithName:@"latitude" fromXML:dealXML];
NSString *longitude = [self contentOfFirstTagWithName:@"longitude" fromXML:dealXML];
NSString *urlString = [self contentOfFirstTagWithName:@"url" fromXML:dealXML];
NSString *thumbnailUrlString = [self contentOfFirstTagWithName:@"thumbnail" fromXML:dealXML];
}
}
/* Convert special characters like & or ’ to unicode characters. */
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSString *)xml
{
// Replace &, ' and " with &, ' and " respectively.
xml = [xml stringByReplacingOccurrencesOfString:@"&" withString:@"&"];
xml = [xml stringByReplacingOccurrencesOfString:@"'" withString:@"'"];
xml = [xml stringByReplacingOccurrencesOfString:@"""
//
// SimpleXMLParser.h
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 7/1/11.
// Copyright 2011 Viewlity.com. All rights reserved.
//
#import <Foundation/Foundation.h>
@interface SimpleXMLParser : NSObject
+ (NSArray *)dealsArrayFromXML:(NSString *)xml;
/* Convert special characters like & or ’ to unico
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSStri
/* Returns the content of the first encounter of a tag from
+ (NSString *)contentOfFirstTagWithName:(NSString *)tagName fromXML:(NSString *)xml;
@end
Let's extract the deal data from each XML component. We do this using a for-in structure. Note that we convert the objects from the components array to NSString because we know they are NSString objects.
We use the contentOfFirstTagWithName: helper method to extract the content of the tags we are interested in. We want to configure our Table View Cells using the title, subtitle and the thumbnail photo. We are going to use the latitude and longitude to present the deals on the map later. We display the deal details in a UIWebView that opens the deal URL. This is all the information we need.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Fri 18:36 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
Run Stop NearbyDeals › iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Today at 00:02 AM
Project 3
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals › NearbyDeals › SimpleXMLParser.m › +dealsArrayFromXML:
S... › +convertSpecialCharactersToUnicodeInXML:
//
// SimpleXMLParser.m
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 7/1/11.
// Copyright 2011 Viewlity.com. All rights reserved.
//
#import "SimpleXMLParser.h"
@implementation SimpleXMLParser
+ (NSArray *)dealsArrayFromXML:(NSString *)xml
{
xml = [self convertSpecialCharactersToUnicodeInXML:xml];
NSMutableArray *components = (NSMutableArray *)[xml componentsSeparatedByString:@"</ad>"];
if ([components count] < 2) return nil;
[components removeLastObject];
NSMutableArray *deals = [[NSMutableArray alloc] initWithCapacity:components.count];
for (NSString *dealXML in components)
{
NSString *title = [self contentOfFirstTagWithName:@"title"
fromXML:dealXML];
NSString *subtitle = [self contentOfFirstTagWithName:@"subtitle"
fromXML:dealXML];
NSString *latitude = [self contentOfFirstTagWithName:@"latitude"
fromXML:dealXML];
NSString *longitude = [self contentOfFirstTagWithName:@"longitude"
fromXML:dealXML];
NSString *urlString = [self contentOfFirstTagWithName:@"url"
fromXML:dealXML];
NSString *thumbnailUrlString = [self contentOfFirstTagWithName:@"thumbnail"
fromXML:dealXML];
NSDictionary *dealInfo = [NSDictionary dictionaryWithObjectsAndKeys:
title, @"title",
subtitle, @"subtitle",
latitude, @"latitude",
longitude, @"longitude",
urlString, @"url",
thumbnailUrlString, @"thumbnail", nil];
}
}
/* Convert special characters like & or ’ to unicode characters. */
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSString *)xml
//
// SimpleXMLParser.h
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 7/1/11.
// Copyright 2011 Viewlity.com. All rights reserved.
//
#import <Foundation/Foundation.h>
@interface SimpleXMLParser : NSObject
+ (NSArray *)dealsArrayFromXML:(NSString *)xml;
/* Convert special characters like & or ’ to unico
+ (NSString *)convertSpecialCharactersToUnicodeInXML:(NSStri
/* Returns the content of the first encounter of a tag from
+ (NSString *)contentOfFirstTagWithName:(NSString *)tagName
fromXML:(NSString *)xml;
@end
We create an NSDictionary that will hold all the information about our deal.


Xcode File Edit View Navigate Editor Product Window Help
Fri 18:40 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — SimpleXMLParser.m
Run Stop NearbyDeals iPhone 5.1 Simulator Scheme Breakpoints
Build Succeeded | Today at 00:02 AM
No Issues
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
NearbyDeals NearbyDeals SimpleXMLParser.m +dealsArrayFromXML:
S... +convertSpecialCharactersToUnicodeInXML:
Next, let's add the NSDictionary that contains our deal info to the deals array.
And return the array of deals not before casting it to its non-mutable superclass (NSArray).

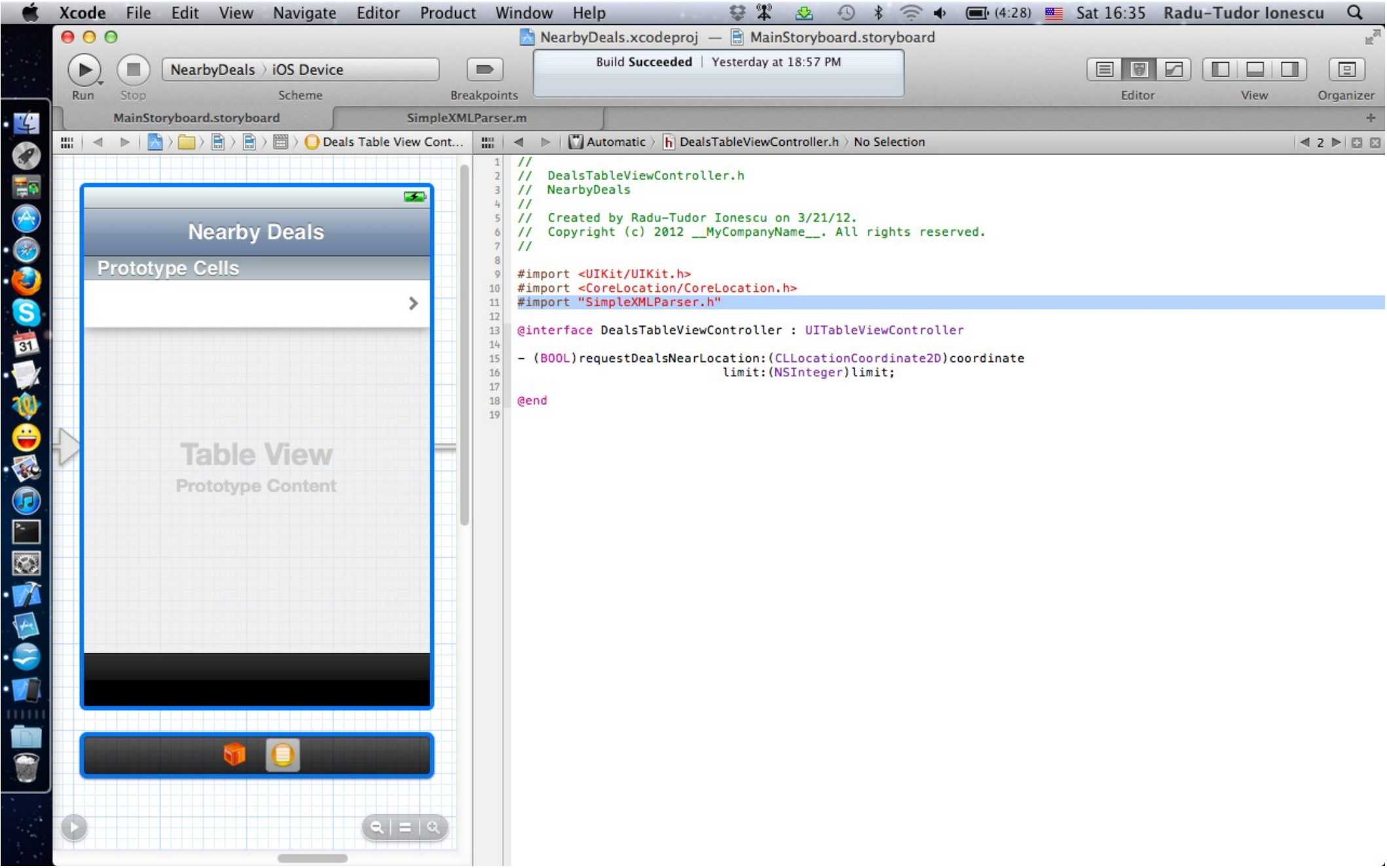
Task 4

Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

1. Switch to the MainStoryboard.storyboard tab in Xcode.
2. Open DealsTableViewController.h in Assistant Editor.
3. We want to use the `SimpleXMLParser dealsArrayFromXML:` class method to obtain an `NSArray` with the nearby deals from the XML file we received from the server.

The first thing to do is to `#import` the `SimpleXMLParser` header file into our Table View Controller so that we can use its methods.

The next screenshot show you how to `#import` this header file.

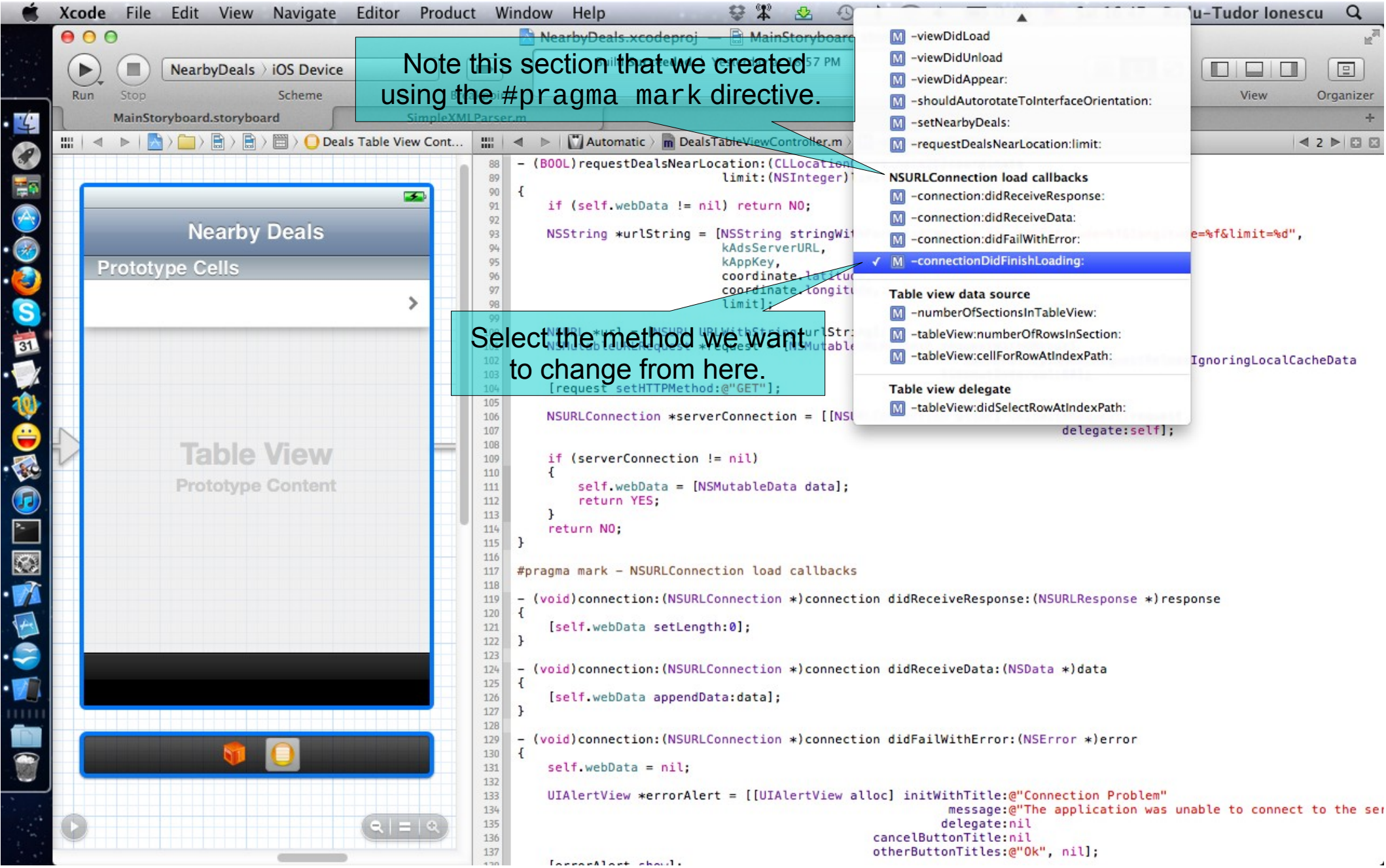

Xcode File Edit View Navigate Editor Product Window Help
Sat 16:35 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iOS Device Scheme Breakpoints
Build Succeeded | Yesterday at 18:57 PM
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
Deals Table View Cont...
Automatic DealsTableViewController.h No Selection
Nearby Deals
Prototype Cells
Table View
Prototype Content
//
// DealsTableViewController.h
// NearbyDeals
//
// Created by Radu-Tudor Ionescu on 3/21/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
#import <CoreLocation/CoreLocation.h>
#import "SimpleXMLParser.h"
@interface DealsTableViewController : UITableViewController
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit;
@end

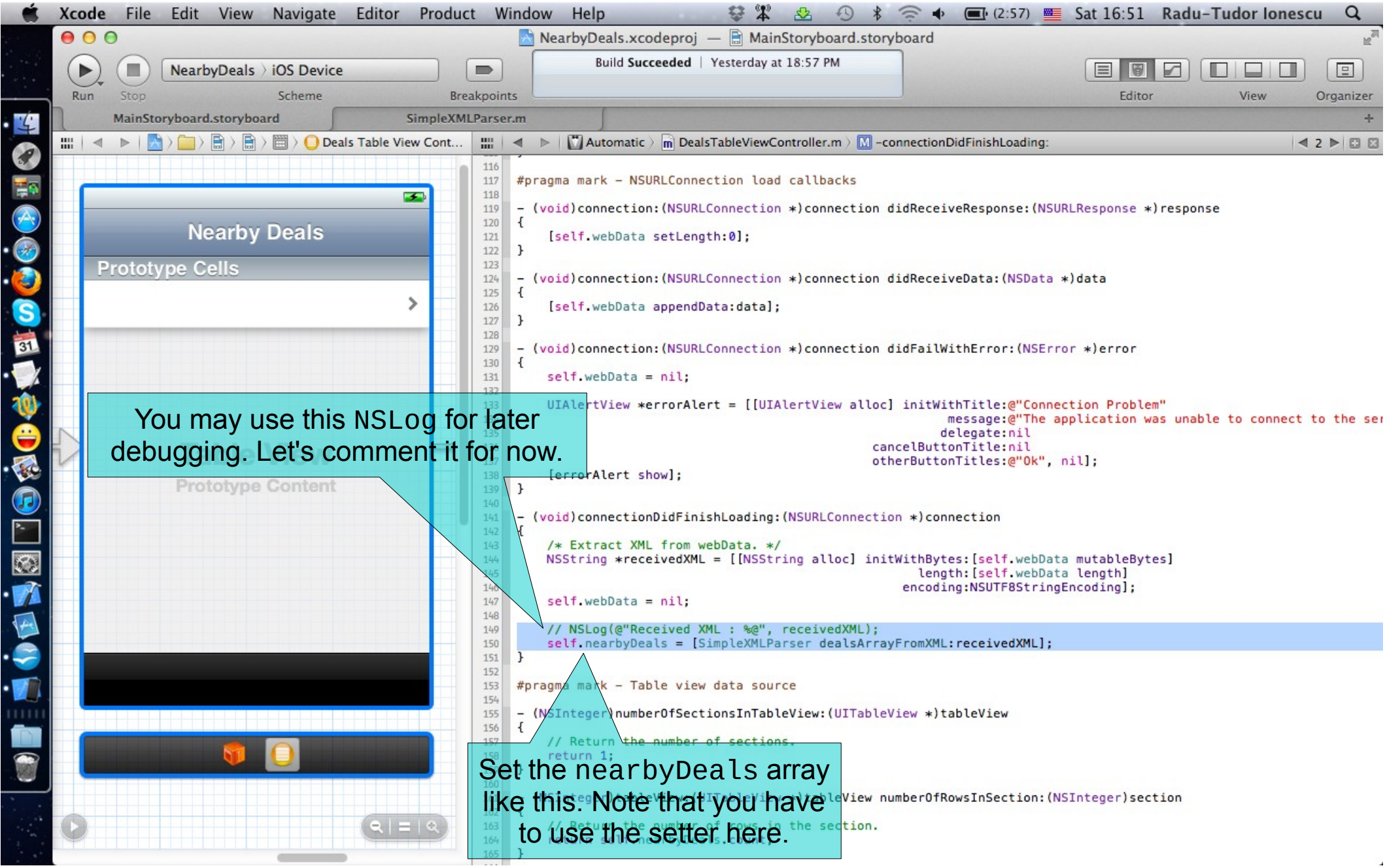
Task 4

Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

4. Open DealsTableViewController.m in Assistant Editor.
5. Scroll to the `connectionDidFinishLoading:` method we implemented earlier. Comment the `NSLog` that prints the XML document to the console.
6. Parse the `receivedXML` using the `dealsArrayFromXML:` class method and store the result into the `nearbyDeals` Model.

The next screenshots show you how to perform these steps.


Note this section that we created using the #pragma mark directive.
Select the method we want to change from here.
-viewDidLoad
-viewDidUnload
-viewDidAppear:
-shouldAutorotateToInterfaceOrientation:
-setNearbyDeals:
-requestDealsNearLocation:limit:
NSURLConnection load callbacks
-connection:didReceiveResponse:
-connection:didReceiveData:
-connection:didFailWithError:
-connectionDidFinishLoading:
Table view data source
-numberOfSectionsInTableView:
-tableView:numberOfRowsInSection:
-tableView:cellForRowAtIndexPath:
Table view delegate
-tableView:didSelectRowAtIndexPath:
Nearby Deals
Prototype Cells
Table View
Prototype Content
#pragma mark - NSURLConnection load callbacks

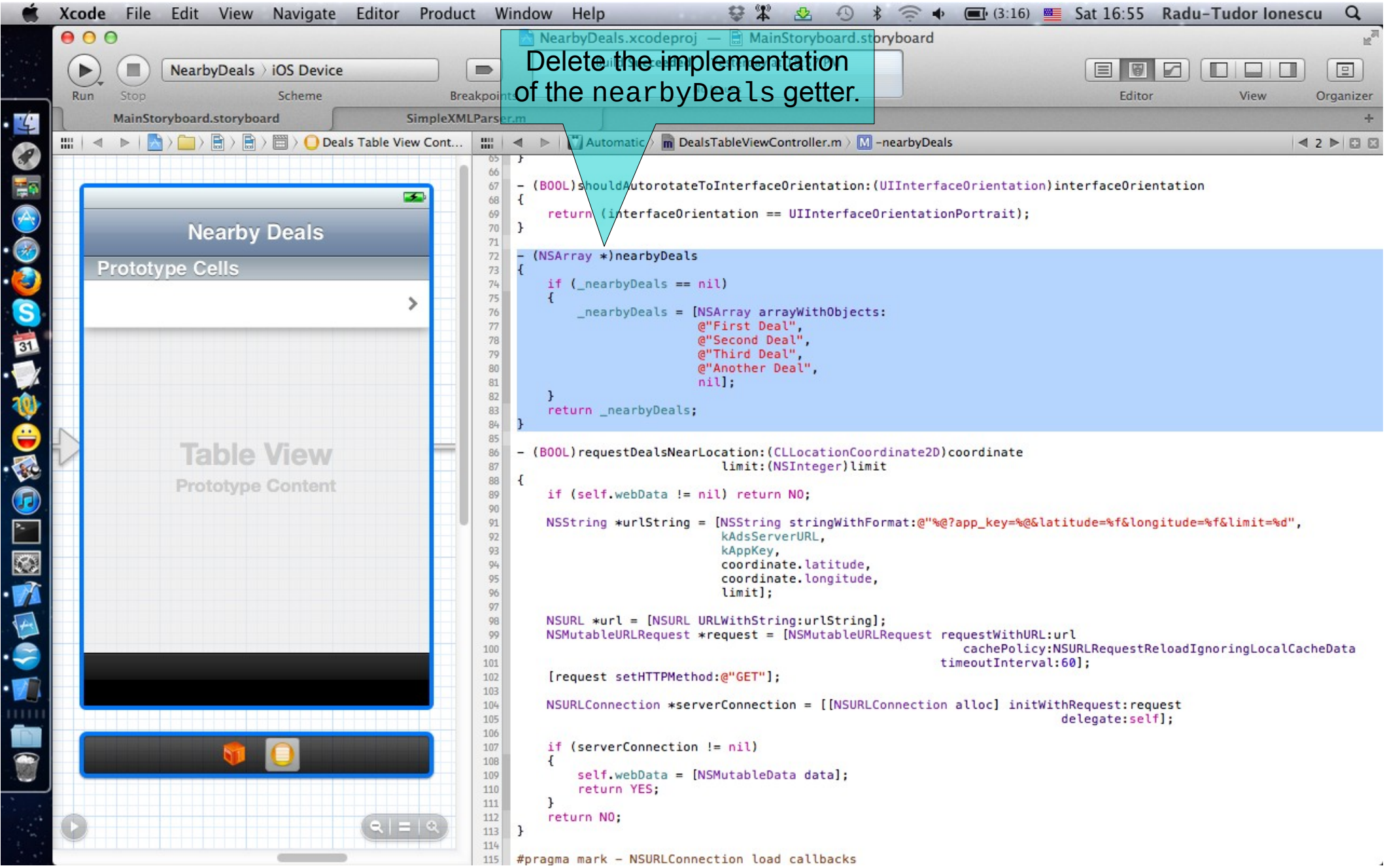

Xcode File Edit View Navigate Editor Product Window Help
Sat 16:51 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Build Succeeded | Yesterday at 18:57 PM
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
Deals Table View Cont...
Automatic DealsTableViewController.m -connectionDidFinishLoading:
Nearby Deals
Prototype Cells
Prototype Content
You may use this NSLog for later debugging. Let's comment it for now.
Set the nearbyDeals array like this. Note that you have to use the setter here.
#pragma mark - NSURLConnection load callbacks
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
[self.webData setLength:0];
}
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
[self.webData appendData:data];
}
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
{
self.webData = nil;
UIAlertView *errorAlert = [[UIAlertView alloc] initWithTitle:@"Connection Problem"
message:@"The application was unable to connect to the ser
delegate:nil
cancelButtonTitle:nil
otherButtonTitles:@"Ok", nil];
[errorAlert show];
}
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
/* Extract XML from webData. */
NSString *receivedXML = [[NSString alloc] initWithBytes:[self.webData mutableBytes]
length:[self.webData length]
encoding:NSUTF8StringEncoding];
self.webData = nil;
// NSLog(@"Received XML : %@", receivedXML);
self.nearbyDeals = [SimpleXMLParser dealsArrayFromXML:receivedXML];
}
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
// Return the number of sections.
return 1;
}
leView numberOfRowsInSection:(NSInteger)section
{
// Return the number of rows in the section.

Task 4

Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

7. We set the `nearbyDeals` `@property` when we receive the XML from the server. We no longer need the lazy instantiation mechanism for `nearbyDeals` because we instantiate it in the `connectionDidFinishLoading:` method. Thus, we can delete the getter implementation.

The next screenshot shows the code that needs to be deleted.


Delete the implementation of the nearbyDeals getter.
Nearby Deals
Prototype Cells
Table View
Prototype Content
- (NSArray *)nearbyDeals
{
if (_nearbyDeals == nil)
{
_nearbyDeals = [NSArray arrayWithObjects:
@"First Deal",
@"Second Deal",
@"Third Deal",
@"Another Deal",
nil];
}
return _nearbyDeals;
}

Task 4

Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

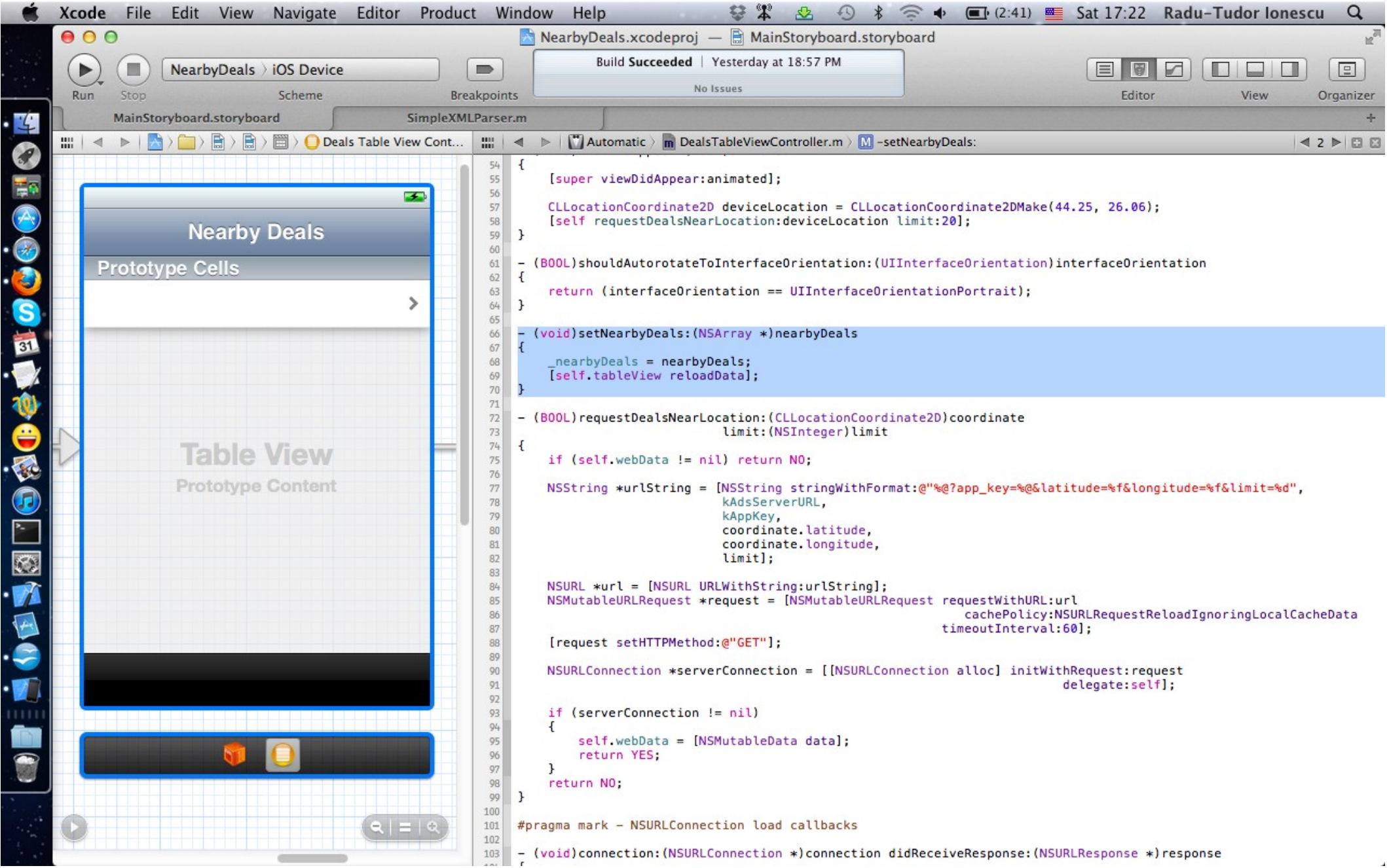
8. We changed our Model in the `connectionDidFinishLoading:` method. We have to let the Table View know that we changed the data so that it has a chance to reload the new data. We do this by sending the `reloadData` message to the `tableView`.

Note that `tableView` is a `@property` inherited by our Table View Controller from `UITableViewController`. It is an outlet of our Table View.

The best place to message the Table View with `reloadData` is the setter of the `nearbyDeals`. If we always instantiate our Model through the setter, we make sure that the Table View always knows about this change as soon as possible.

Let's override the `nearbyDeals` setter with our own implementation and send the `reloadData` to the Table View there.

The next screenshot shows you how to do this.


Xcode File Edit View Navigate Editor Product Window Help
Sat 17:22 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iOS Device Scheme Breakpoints
Build Succeeded | Yesterday at 18:57 PM
No Issues
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
Deals Table View Cont...
Automatic DealsTableViewController.m -setNearbyDeals:
Nearby Deals
Prototype Cells
Table View
Prototype Content
{
[super viewDidAppear:animated];
CLLocationCoordinate2D deviceLocation = CLLocationCoordinate2DMake(44.25, 26.06);
[self requestDealsNearLocation:deviceLocation limit:20];
}
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
- (void)setNearbyDeals:(NSArray *)nearbyDeals
{
_nearbyDeals = nearbyDeals;
[self.tableView reloadData];
}
- (BOOL)requestDealsNearLocation:(CLLocationCoordinate2D)coordinate
limit:(NSInteger)limit
{
if (self.webData != nil) return NO;
NSString *urlString = [NSString stringWithFormat:@"%@?app_key=%@&latitude=%f&longitude=%f&limit=%d",
kAdsServerURL,
kAppKey,
coordinate.latitude,
coordinate.longitude,
limit];
NSURL *url = [NSURL URLWithString:urlString];
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:url
cachePolicy:NSURLRequestReloadIgnoringLocalCacheData
timeoutInterval:60];
[request setHTTPMethod:@"GET"];
NSURLConnection *serverConnection = [[NSURLConnection alloc] initWithRequest:request
delegate:self];
if (serverConnection != nil)
{
self.webData = [NSMutableData data];
return YES;
}
return NO;
}
#pragma mark - NSURLConnection load callbacks
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response

Task 4

Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

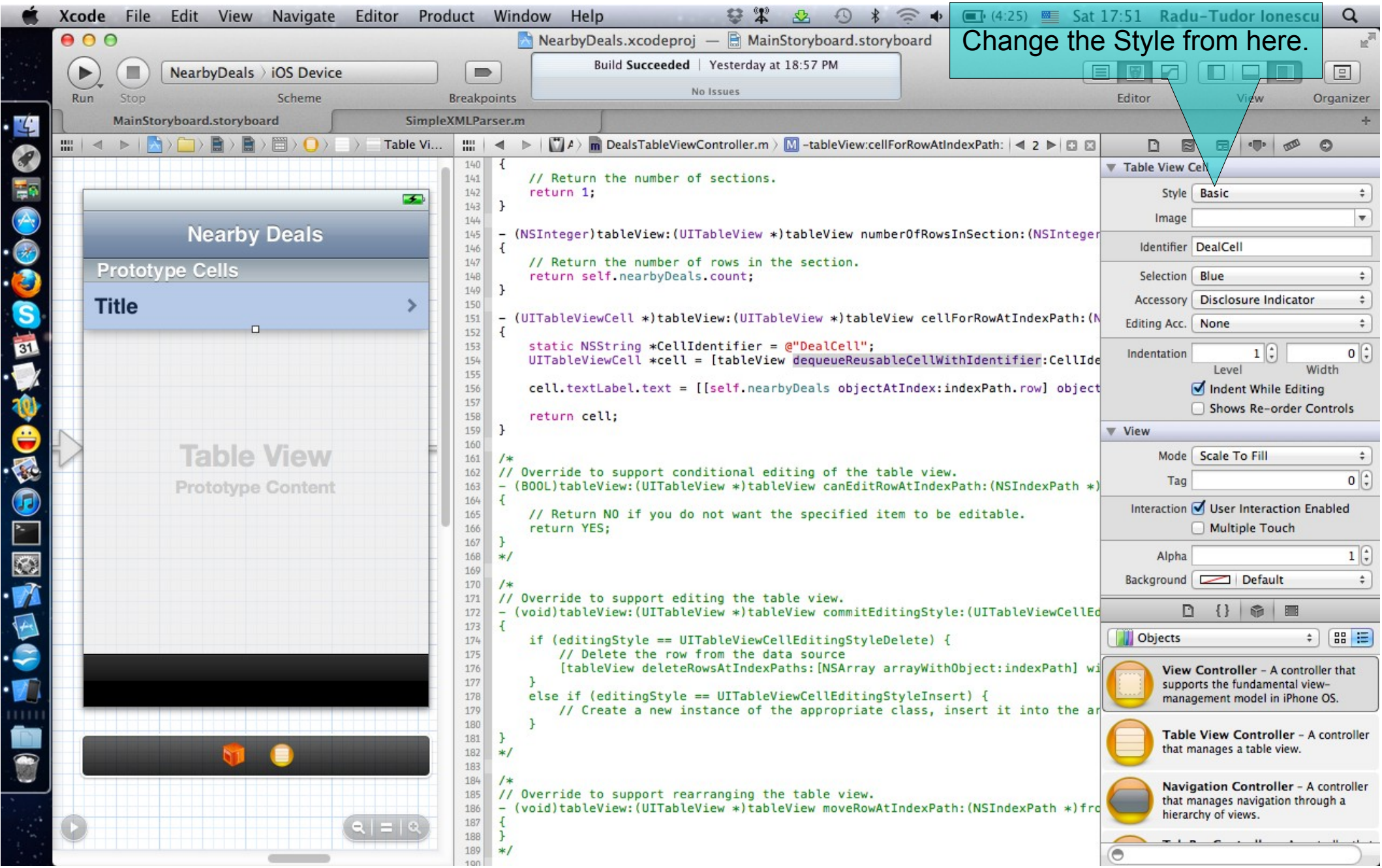
9. Let's show up the Utilities area.

10. Click on the Prototype Cell in Interface Builder to see its attributes in Inspector.

11. Change the Prototype Cell's Style to Basic in Attributes Inspector.

Look on the next screenshot for help.

12. Hide back the Utilities area.

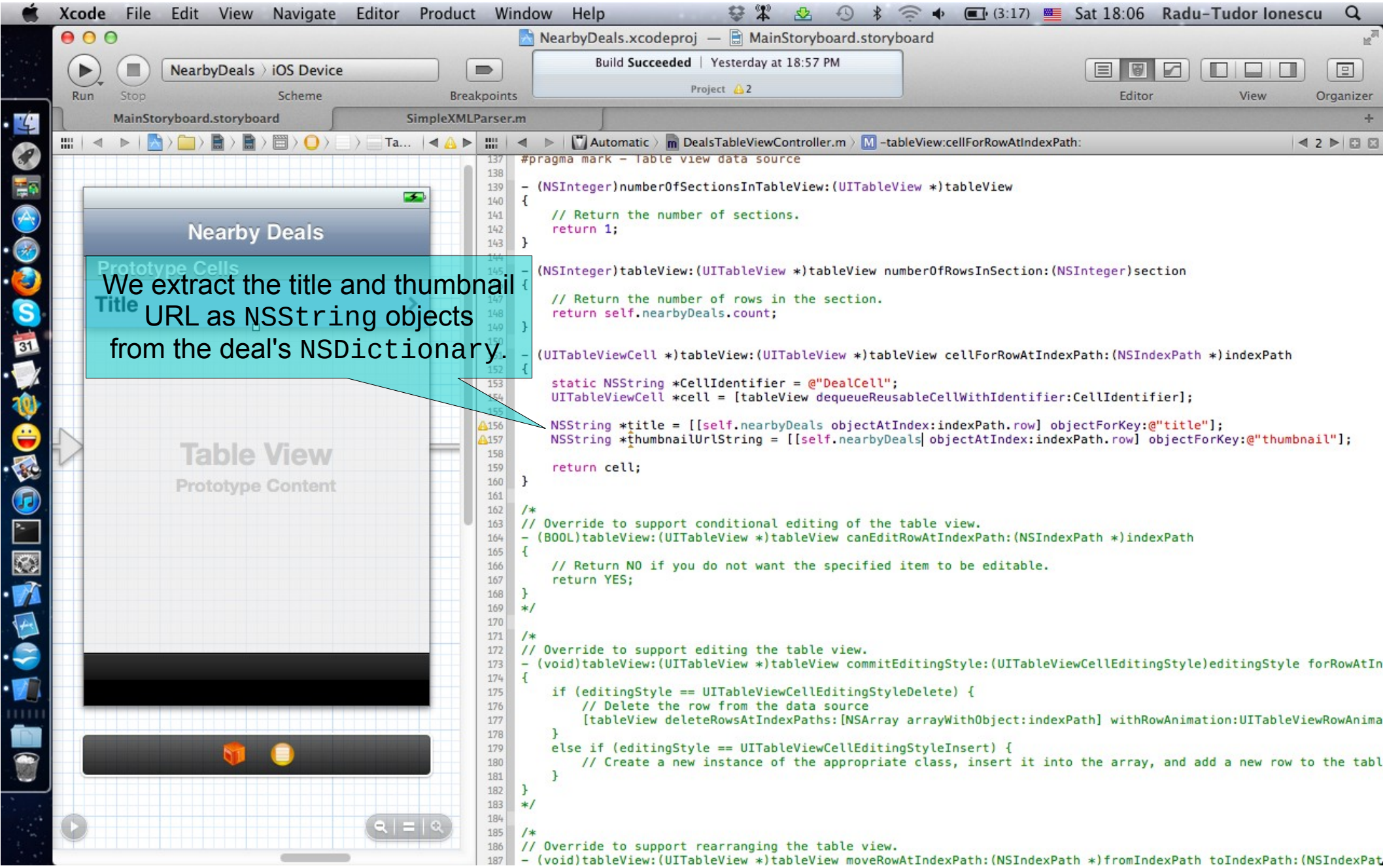

Change the Style from here.
Nearby Deals
Prototype Cells
Title
Table View
Prototype Content
Table View Cell
Style Basic
Image
Identifier DealCell
Selection Blue
Accessory Disclosure Indicator
Editing Acc. None
Indentation 1 0
Level Width
Indent While Editing
Shows Re-order Controls
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Objects
View Controller – A controller that supports the fundamental view-management model in iPhone OS.
Table View Controller – A controller that manages a table view.
Navigation Controller – A controller that manages navigation through a hierarchy of views.

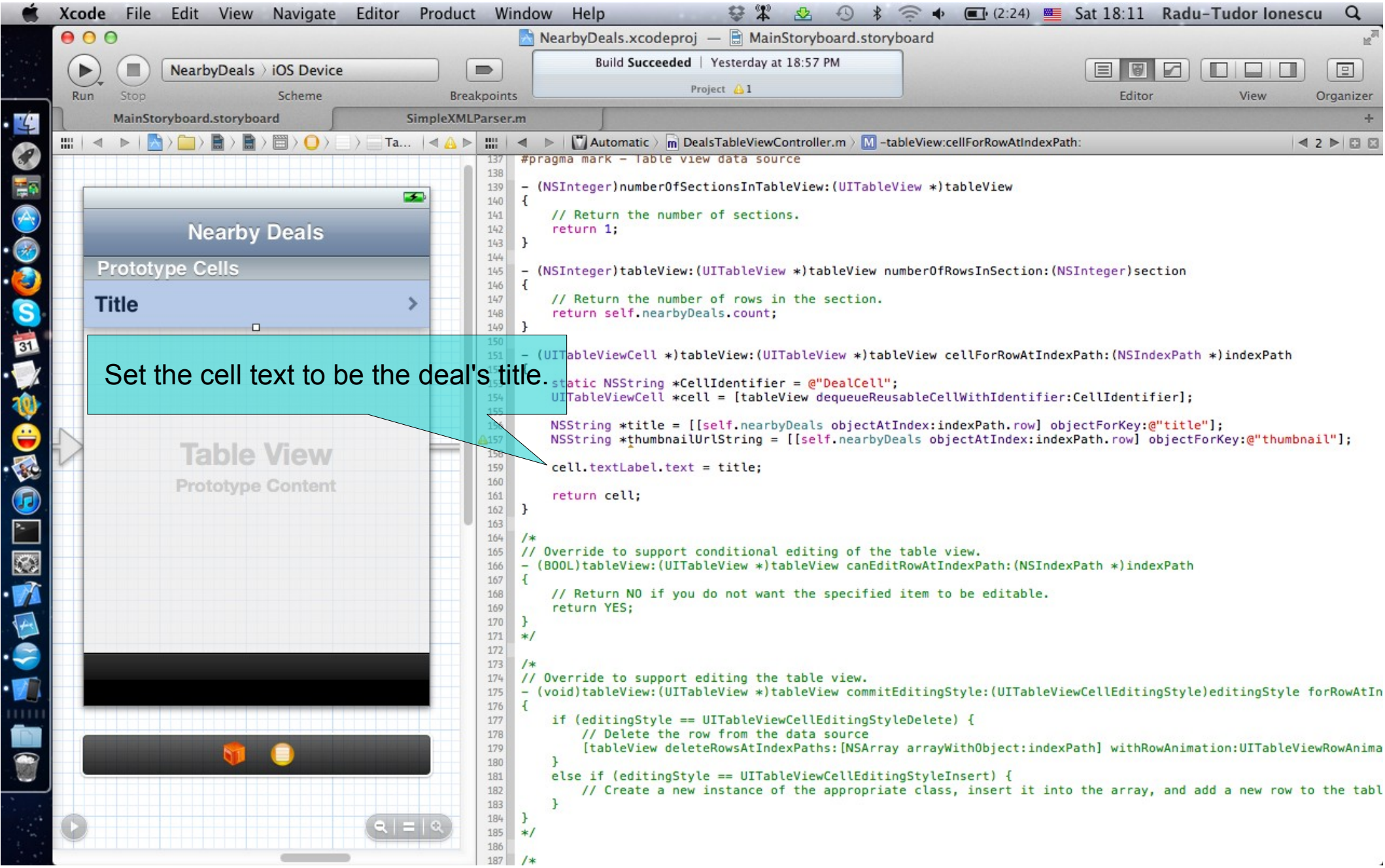
Task 4

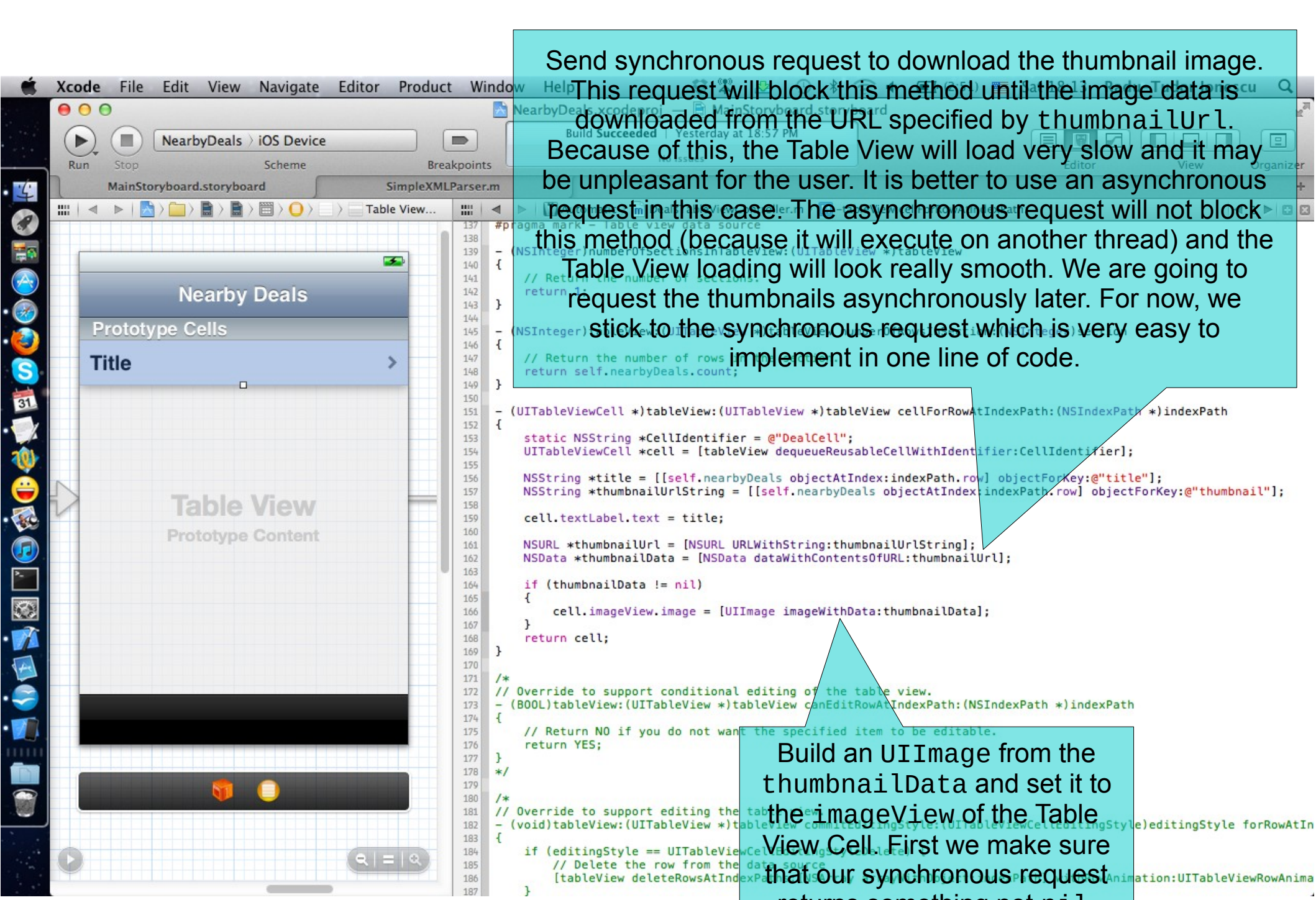
Task: Use the `SimpleXMLParser` class methods to parse the XML with nearby deals and display them in your Table View.

13. Re-implement the `tableView:cellForRowAtIndexPath:` method to use the data from the new Model (that we received from the GeoAds+ server). Each Table View Cell will display information about a deal. For now, we want to present the deal title and its thumbnail photo. We extract this information from the `NSDictionary` at `indexPath.row` inside the `nearbyDeals` array.

The next slides will guide you through the re-implementation of this method.


We extract the title and thumbnail URL as NSString objects from the deal's NSDictionary.
Nearby Deals
Prototype Cells
Title
Table View
Prototype Content
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
// Return the number of sections.
return 1;
}
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
// Return the number of rows in the section.
return self.nearbyDeals.count;
}
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
static NSString *CellIdentifier = @"DealCell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
NSString *title = [[self.nearbyDeals objectAtIndex:indexPath.row] objectForKey:@"title"];
NSString *thumbnailUrlString = [[self.nearbyDeals objectAtIndex:indexPath.row] objectForKey:@"thumbnail"];
return cell;
}
/*
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
{
// Return NO if you do not want the specified item to be editable.
return YES;
}
*/
/*
// Override to support editing the table view.
- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIn
{
if (editingStyle == UITableViewCellEditingStyleDelete) {
// Delete the row from the data source
[tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:UITableViewRowAnima
}
else if (editingStyle == UITableViewCellEditingStyleInsert) {
// Create a new instance of the appropriate class, insert it into the array, and add a new row to the tabl
}
}
*/
/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:(NSIndexPath *)fromIndexPath toIndexPath:(NSIndexPat


Xcode File Edit View Navigate Editor Product Window Help
Sat 18:11 Radu-Tudor Ionescu
NearbyDeals.xcodeproj — MainStoryboard.storyboard
Run Stop NearbyDeals iOS Device Scheme Breakpoints
Build Succeeded | Yesterday at 18:57 PM
Project 1
Editor View Organizer
MainStoryboard.storyboard
SimpleXMLParser.m
Automatic DealsTableViewController.m -tableView:cellForRowAtIndexPath:
Nearby Deals
Prototype Cells
Title
Table View
Prototype Content
Set the cell text to be the deal's title.
#pragma mark - Table view data source
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
// Return the number of sections.
return 1;
}
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
// Return the number of rows in the section.
return self.nearbyDeals.count;
}
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
static NSString *CellIdentifier = @"DealCell";
UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
NSString *title = [[self.nearbyDeals objectAtIndex:indexPath.row] objectForKey:@"title"];
NSString *thumbnailUrlString = [[self.nearbyDeals objectAtIndex:indexPath.row] objectForKey:@"thumbnail"];
cell.textLabel.text = title;
return cell;
}
/*
// Override to support conditional editing of the table view.
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath
{
// Return NO if you do not want the specified item to be editable.
return YES;
}
*/
/*
// Override to support editing the table view.
- (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIn
{
if (editingStyle == UITableViewCellEditingStyleDelete) {
// Delete the row from the data source
[tableView deleteRowsAtIndexPaths:[NSArray arrayWithObject:indexPath] withRowAnimation:UITableViewRowAnima
}
else if (editingStyle == UITableViewCellEditingStyleInsert) {
// Create a new instance of the appropriate class, insert it into the array, and add a new row to the tabl
}
}
*/
/*


Send synchronous request to download the thumbnail image. This request will block this method until the image data is downloaded from the URL specified by thumbnailUrl. Because of this, the Table View will load very slow and it may be unpleasant for the user. It is better to use an asynchronous request in this case. The asynchronous request will not block this method (because it will execute on another thread) and the Table View loading will look really smooth. We are going to request the thumbnails asynchronously later. For now, we stick to the synchronous request which is very easy to implement in one line of code.
Nearby Deals
Prototype Cells
Title
Table View
Prototype Content
Build an UIImage from the thumbnailData and set it to the imageView of the Table View Cell. First we make sure that our synchronous request returns something not nil.

Assignment 1

Assignment: Adjust the GeoAds+ URL string that is constructed inside the `requestDealsNearLocation:limit:` method in order to obtain nearby deals only for Restaurants and Bars.

Hint: Look at the request to GeoAds+ API we made from Safari.

Assignment 2

Assignment: Add the deal's subtitle to the Table View Cell.

Hint: You have to change the Prototype Cell Style to Subtitle and set the cell's `detailTextLabel` programmatically.

Congratulations!