LocalSearch App

Search for Nearby Points of Interest Using Local Search

Introduced in iOS 7, the Search API (i.e MKLocalSearch) allows iOS developers to search for points of interest and display them on maps. App developers can use this API to perform searches for locations, which can be name, address, or type, such as coffee or pizza.

The use of MKLocalSearch is very similar to the MKDirections API covered in the previous practice (GetDirection). You'll first need to create an MKLocalSearchRequest object that bundles your search query. You can also specify the map region to narrow down the search result. You then use the configured object to initialize an MKLocalSearch object and perform the search.

The search is performed remotely in an asynchronous way. Once Apple returns the search result (as an MKLocalSearchResponse object) to your app, the complete handler will be executed. In general, you'll need to parse the response object and display the search results on the map.

Local Search Demo App

There is no better way to understand local search than working on a demo project. Again we will not start from scratch but build on top of the previous project to add a Nearby feature. When you tap the Nearby button, the app searches for nearby restaurants and pins the places on the map.

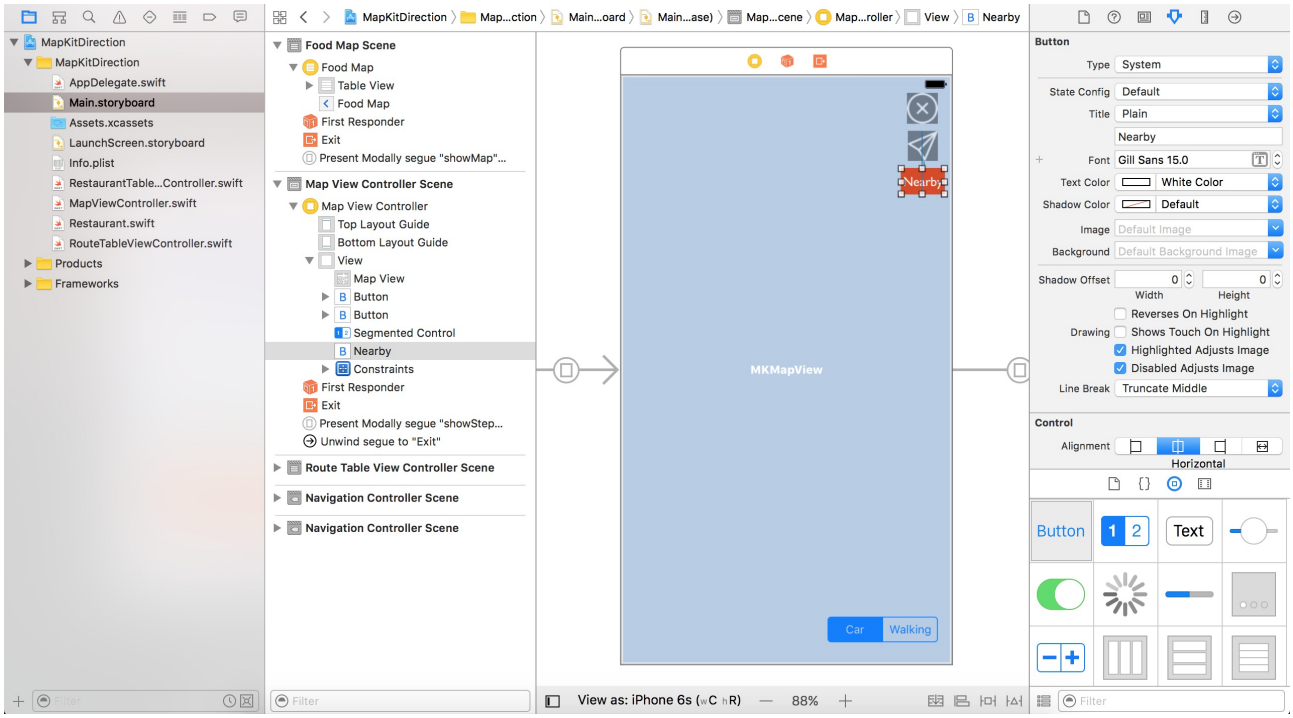
To start with, first download the Xcode project template from (LocalSearchStarter.zip). Unzip it and open the

MapKitDirection project.

Note: The starter project is exactly the same as the final project of the MapKit Direction app.

Adding a Nearby Button in Storyboard

First go to Main.storyboard and add a button item to the map view. Name the button Nearby. Optionally you can change the font style and the background color of the button. After that, click the Issues button in the layout bar and select Add Missing Constraints.



Search Nearby Restaurants and Adding annotations

Once you added the button, open the MapViewController.swift file. We will create an action method called showNearby for the Nearby button. In the implementation, we will search for nearby restaurants and pin the results on the map.

Insert the following code snippet in the class:

```
func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) ->
MKAnnotationView? {
    let identifier = "MyPin"
    if annotation.isKind(of: MKUserLocation.self) {
        return nil
    }

    // Reuse the annotation if possible
    var annotationView:MKPinAnnotationView? =
mapView.dequeueReusableAnnotationView(withIdentifier: identifier) as?
MKPinAnnotationView
    if annotationView == nil {
        annotationView = MKPinAnnotationView(annotation: annotation,
                                             reuseIdentifier: identifier)
        annotationView?.canShowCallout = true
    }
    // Pin color customization based on the type of annotation
    if let currentPlacemarkCoordinate = currentPlacemark?.location?.coordinate {
        if currentPlacemarkCoordinate.latitude == annotation.coordinate.latitude
        && currentPlacemarkCoordinate.longitude == annotation.coordinate.longitude {
            let leftIconView = UIImageView(frame: CGRect.init(x: 0, y: 0, width: 53,
            height: 53))
            leftIconView.image = UIImage(named: restaurant.image)
            annotationView?.leftCalloutAccessoryView = leftIconView
            // Pin color customization
            if #available(iOS 9.0, *) {
                annotationView?.pinTintColor = UIColor.orange
            }

        } else {

            // Pin color customization
            if #available(iOS 9.0, *) {
                annotationView?.pinTintColor = UIColor.red
            }

        }

    }

    annotationView?.rightCalloutAccessoryView = UIButton(type:
    UIButtonType.detailDisclosure)
```

```
    return annotationView
}
```

To perform a local search, here are the two things you need to do:

Specify your search parameters in an MKLocalSearchRequest object. You are allowed to specify the search criteria in natural language by using the naturalLanguageQuery parameter. For example, if you want to search for a nearby cafe, you can specify cafe in the search parameter. Since we want to search for similar types of restaurants, we specify

restaurant.type in the query.

Initiate the local search by creating an MKLocalSearch object with the search parameters. An MKLocalSearch object is used to initiate a map-based search operation and delivers the results back to your app asynchronously.

In the showNearby method, we lookup the nearby restaurants that are of the same type (e.g. Italian). Furthermore, we specify the current region of the map view as the search region.

```
@IBAction func showNearby(sender: UIButton) {
    let searchRequest = MKLocalSearchRequest()
    searchRequest.naturalLanguageQuery = restaurant.type
    searchRequest.region = mapView.region
    let localSearch = MKLocalSearch(request: searchRequest)
    localSearch.start { (response, error) -> Void in
        guard let response = response else {
            if let error = error {
                print(error)
            }
            return
        }
        let mapItems = response.mapItems
        var nearbyAnnotations: [MKAnnotation] = []
        if mapItems.count > 0 {
            for item in mapItems {
```

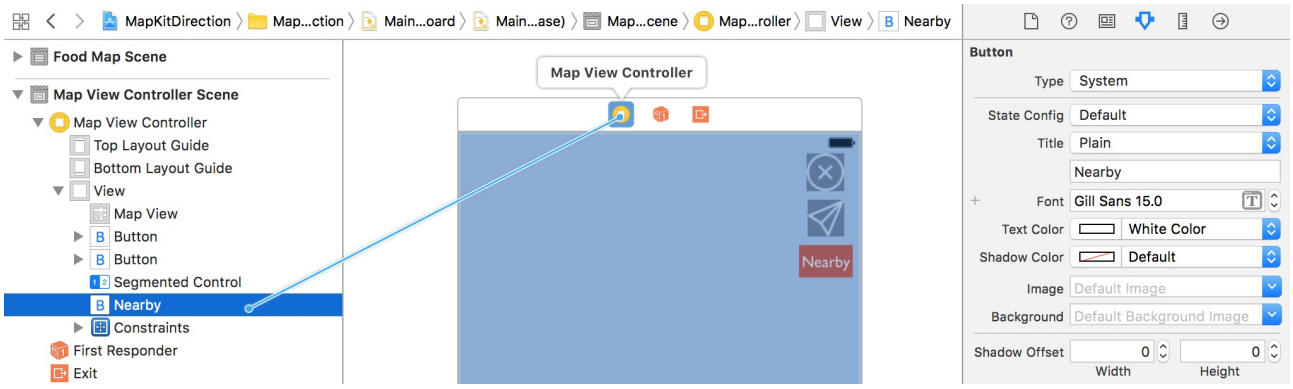
```
            // Add annotation
            let annotation = MKPointAnnotation()
            annotation.title = item.name
            annotation.subtitle = item.phoneNumber
            if let location = item.placemark.location {
              annotation.coordinate = location.coordinate
            }
            nearbyAnnotations.append(annotation)
          }
        }
        self.mapView.showAnnotations(nearbyAnnotations, animated: true)
    }
}
```

We then initialize the search by creating the MKLocalSearch object and invoking the start(completionHandler:) method. When the search completes, the closure will be called and

the results are delivered as an array of MKMapItem . In the body of the closure, we loop through the items (i.e. nearby restaurants) and highlight them on the map using annotations. To pin multiple annotations on maps, you call the showAnnotations method and pass it the array of

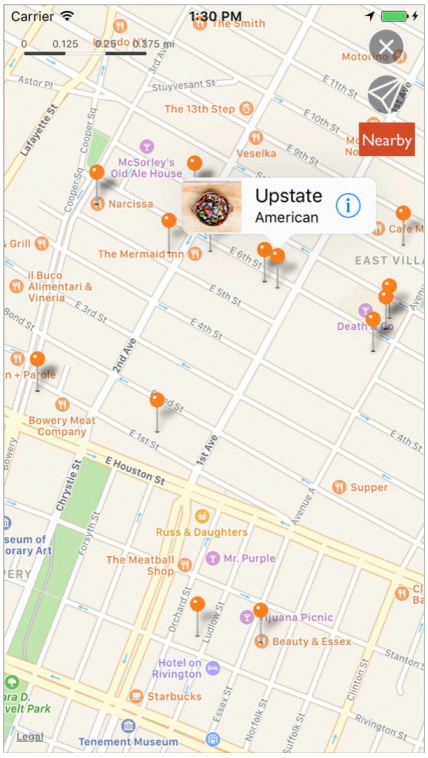
MKAnnotation objects to pin.

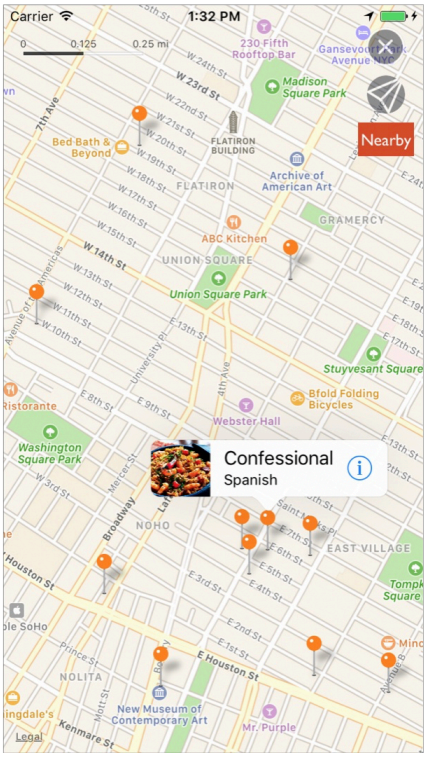
Now go to the storyboard and connect the Nearby button with the showNearby method. Simply control-drag from the Nearby button to the view controller icon in the scene dock and select the showNearbyWithSender: action method.



Testing the Demo App

Now hit the Run button to compile and run your app. Select a restaurant to bring up the map view. Tap the Nearby button and the app should show you the nearby restaurants.





With just a few lines of code, you took your Map app to the next level. If you're going to embed a map within your app, try to explore the local search API.

Modifying the Pin Color for the Nearby Restaurants

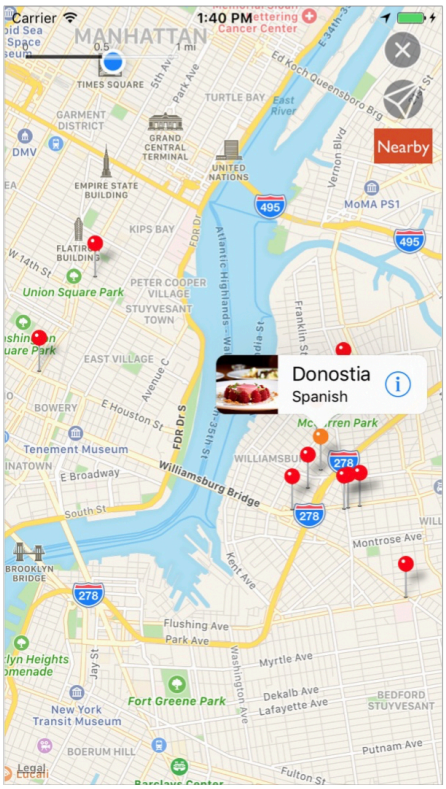
With the current implementation, all the pins are of the same color. If you tap a nearby restaurant, the callout bubble still displays the current restaurant image. How can we make it better? You probably want to differentiate the nearby restaurants with the current selection by using a different pin color. To do so, you can modify the mapView(_:viewFor:) method, which is responsible for the appearance of an annotation.

```
func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) ->
MKAnnotationView? {
    let identifier = "MyPin"
    if annotation.isKind(of: MKUserLocation.self) {
        return nil
    }

    // Reuse the annotation if possible
    var annotationView:MKPinAnnotationView? =
        mapView.dequeueReusableAnnotationView(withIdentifier: identifier) as?
        MKPinAnnotationView
    if annotationView == nil {
        annotationView = MKPinAnnotationView(annotation: annotation,
                                             reuseIdentifier: identifier)
        annotationView?.canShowCallout = true
    }

    if let currentPlacemarkCoordinate = currentPlacemark?.location?.coordinate {
        if currentPlacemarkCoordinate.latitude == annotation.coordinate.latitude &&
            currentPlacemarkCoordinate.longitude == annotation.coordinate.longitude {
                let leftIconView = UIImageView(frame: CGRect.init(x: 0, y: 0, width: 53,
                                               height: 53))
                leftIconView.image = UIImage(named: restaurant.image)
```

```
            annotationView?.leftCalloutAccessoryView = leftIconView
            // Pin color customization
            if #available(iOS 9.0, *) {
                annotationView?.pinTintColor = UIColor.orange
            }

        } else {

            // Pin color customization
            if #available(iOS 9.0, *) {
                annotationView?.pinTintColor = UIColor.red
            }

        }

    }

    annotationView?.rightCalloutAccessoryView = UIButton(type:
                                                         UIButtonType.detailDisclosure)
    return annotationView
}
```

We have modified the method a bit by adding a condition block. The code checks if the annotation, which is about to display, is the same as the current placemark. In other words, we check if the coordinate of the annotation is equal to the coordinate of the selected restaurant. If the result is positive, we display the restaurant image in the callout bubble and keep the pin color to orange. Otherwise, for those nearby restaurants, we set the pin color to red.

Now run the project and try out the Nearby feature again. The pin color of the nearby restaurants is now in red.


Carrier
1:40 PM
MANHATTAN
Nearby
Donostia
Spanish
Union Square Park
Williamsburg Bridge
Fort Greene Park
NEW YORK


Carrier
1:41 PM
Homei
Cafe
Nearby
Whitney Museum of American Art
Union Square Park
Washington Square Park
Hudson River Park - Pier 40
Tenement Museum
NEW YORK

***