



Location And Maps

Sisoft Technologies Pvt Ltd

SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad

Website: www.sisoft.in Email: info@sisoft.in

Phone: +91-9999-283-283



Core Location

- Core Location framework is used to determine the current latitude and longitude of a device
- To use the features of the Core Location framework, you must link your app to CoreLocation.framework in your Xcode project
- To access the classes and headers of the framework, include an `#import <CoreLocation/CoreLocation.h>` statement at the top of any relevant source files.



Core Location

- There are three technologies that core location uses :
 - **GPS**

Reads microwave signals from multiple satellites to determine the current location
 - **Cell Tower Triangulation**

Determine the current location by calculation based on location of cell towers in iPhone's range.
 - **Wi-Fi positioning Service (WPS)**

Uses IP address from iPhone's Wi-Fi connection by referencing database of service providers and areas they service



CLLocationManager

- This class defines the interface for configuring the delivery of location-related events to your application
- To interact with core location we need to create an instance of Location Manager
 - CLLocationManager *locationManager
= [[CLLocationManager alloc] init];
- When you are ready to start polling for location, you tell the location manager to start
 - [locationManager startUpdatingLocation];



Setting Desired Accuracy

- The accuracy is set using **CLLocationAccuracy** Value, a type that's defined as double
 - `kCLLocationAccuracyBest`
 - `kCLLocationAccuracyNearestTenMeters`
 - `kCLLocationAccuracyHundredMetres`
 - `kCLLocationAccuracyKilometers`
 - `kCLLocationAccuracyThreeKilometers`
- **Setting Delegate and Accuracy:**

```
locationManager.delegate=self;  
locationManager.desiredAccuracy= CLLocationAccuracyBest
```



Setting Distance Filter

- By specifying distance filter you can tell Location manager not to notify for every change in location but to only notify you when location changes by certain amount
- Specifying a distance filter of 1000 tells location manager not to notify the delegate until the iPhone has moved at least 1000 meters from previously reported position
- For example:

locationManager.distanceFilter=1000.0f;



Location Object (CLLocation)

- Represents the location data generated by a CLLocationManager object
- The latitude and longitude are stored in property called coordinate
 - CLLocationDegrees latitude=theLocation.coordinate.latitude
 - CLLocationDegrees longitude=theLocation.coordinate.longitude
- The CLLocation object also has a property called altitude that tell you how many metres above the sea level you are:
 - CLLocationDistance altitude=theLocation.altitude



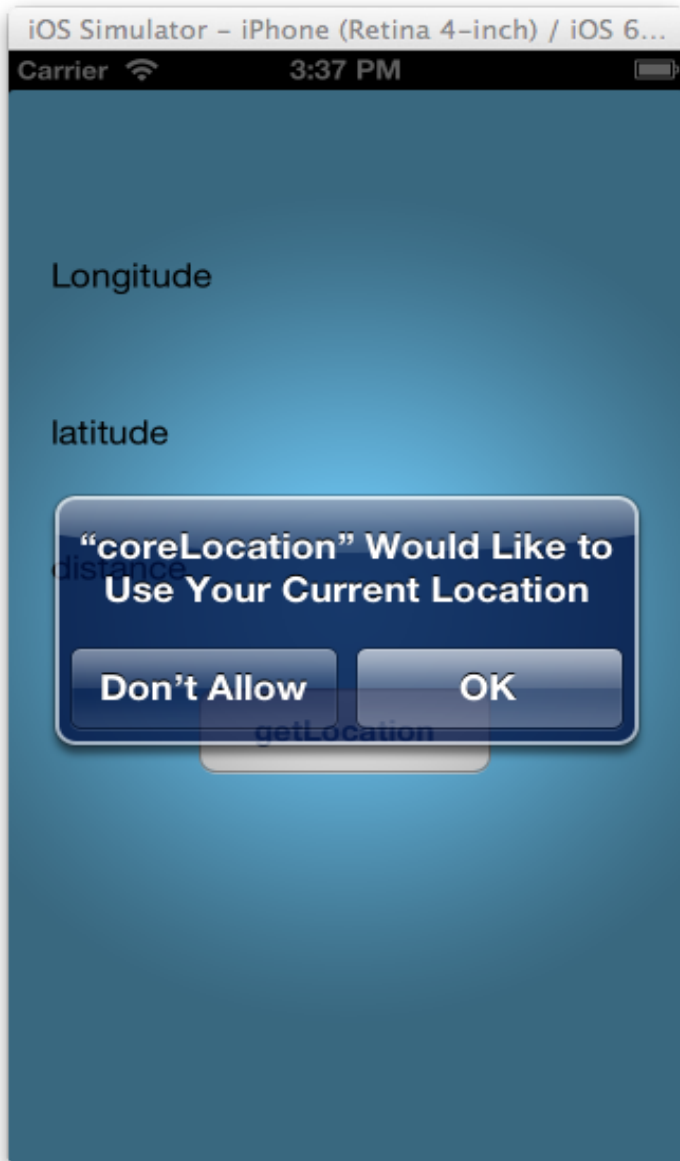
CLLocationManagerDelegate protocol

- The location manager must confirm to CLLocationManagerDelegate protocol, which defines two methods, both are optional
- One method is called by location manager when it has determined current location or detects any change in location:

locationManager:didUpdateToLocation:fromLocation

- Other method is called when location manager encounters an error:

locationManager:didFailWithError





MapKit

- Maps are provided by the Map Kit framework, which supports both the display and annotation of maps
- Plan
 - Adding MapKit to your iPhone Application
 - Setting and zooming the Position
 - Annotations
 - Geocoding

Adding Mapkit



The image shows a screenshot of the Xcode interface. On the left, a storyboard titled "My Maps" contains a large blue rectangle labeled "MKMapView". A grey arrow points from the left edge of the storyboard to the "MKMapView" label. On the right, the "Map View" Inspector is open, showing various configuration options. An orange arrow points from the "Shows User Location" checkbox in the Behavior section to the text "Enable User Location". Another orange arrow points from the "Map View" entry in the Objects list at the bottom to the "MKMapView" label in the storyboard, with the text "Drag to add Map View" next to it.

Map View

Type:

Behavior

- Shows User Location
- Allows Zooming
- Allows Scrolling

Mode:

Tag:

Interaction

- User Interaction Enabled
- Multiple Touch

Alpha:

Background:

Drawing

- Opaque Hidden
- Clears Graphics Context
- Clip Subviews
- Autoresize Subviews

Stretching

X: Y:

Width: Height:

Objects

- Web View** - Displays embedded web content and enables content navigation.
- Map View** - Displays maps and provides an embeddable interface to navigate map content.
- Scroll View** - Provides a mechanism to display content that is larger than the size of the application's window.

Adding Framework



A screenshot of the Xcode interface showing the project settings for 'MyLocationMap'. The left sidebar shows the 'PROJECT' and 'TARGETS' sections, with 'MyLocationMap' selected under 'TARGETS'. The main area displays the 'Build Phases' tab, which is divided into sections: 'Target Dependencies (0 items)', 'Compile Sources (3 items)', 'Link Binary With Libraries (5 items)', and 'Copy Bundle Resources (5 items)'. The 'Link Binary With Libraries' section is expanded, showing a list of frameworks: MapKit.framework, CoreLocation.framework, UIKit.framework, Foundation.framework, and CoreGraphics.framework. At the bottom of this list, there are '+' and '-' icons and the text 'Drag to reorder frameworks'. The top of the main area has tabs for 'Summary', 'Info', 'Build Settings', and 'Build Phases', with 'Build Phases' being the active tab. A search bar is visible in the top right corner of the main area.

MKMapViewDelegate protocol



- The MKMapViewDelegate protocol defines a set of optional methods that you can use to receive map-related update messages
- The MKMapViewDelegate protocol have methods for:
 - Responding to Map Position Changes
 - Loading the Map Data
 - Tracking the User Location
 - Managing Annotation Views
 - Managing the Display of Overlays



Map Region to Display

- A structure that defines which portion of the map to display.
- Set the **region** property (MKCoordinateRegion)
- Define the **center** point and a **span**
- **Span** defines the vertical & horizontal distance to display (zoom)

```
{ MKCoordinateRegion region;  region.center.latitude = [eventVenue.latitude  
doubleValue];  region.center.longitude = [eventVenue.longitude doubleValue];  
region.span.latitudeDelta = 0.0039;  region.span.longitudeDelta = 0.0034; //zoom ~.3  
miles  mapView.region = region;  
}
```



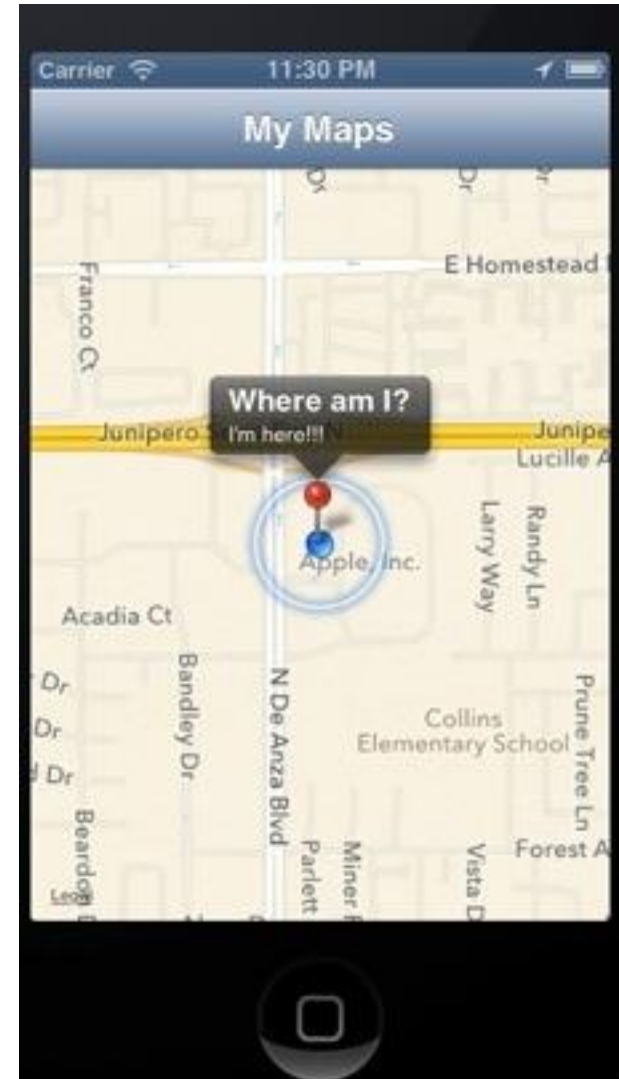
Annotations

- Annotations offer a way to highlight specific coordinates on the map and provide additional information about them
- In order to display an annotation on a map, your app must provide two distinct objects:
 - An object that conforms to the MKAnnotation protocol and manages the data for the annotation(annotation object)
 - A view used to draw the visual representation of the annotation(annotation view)

MKPointAnnotation



- `// Add an annotation`
 - `MKPointAnnotation *point = [[MKPointAnnotation alloc] init];`
 - `point.coordinate = userLocation.coordinate;` `point.title = @"Where am I?";`
 - `point.subtitle = @"I'm here!!!";` `[self.mapView addAnnotation:point];`





CLGeocoder

- The CLGeocoder class provides services for converting between a GPS coordinate and the user-readable address of that coordinate
- By specifying the latitude and longitude of a given location, you can use CLGeocoder to find a user-readable address
- The result (i.e. the address) returned by CLGeocoder is saved in a CLPlacemark object