CORE LOCATION

TUTORIAL ON

USING THE CORE LOCATION FRAMEWORK

Shubham Singh
@shubham_iosdev

CORE LOCATION

INTRODUCTION

- Core Location framework provides APIs for tracking user locations, and you will need to use it when you have to work with maps.
- You can use it to determine the user's geographic location, altitude, track large or small changes in the user's location.
- In this tutorial, we will be creating a Location Manager class that uses the APIs of the Core Location framework.
- Creating a separate Location Manager prevents tight coupling of code, and it becomes a lot more reusable.

Note- I recommend you to try out the CoreLocation framework in a real device.

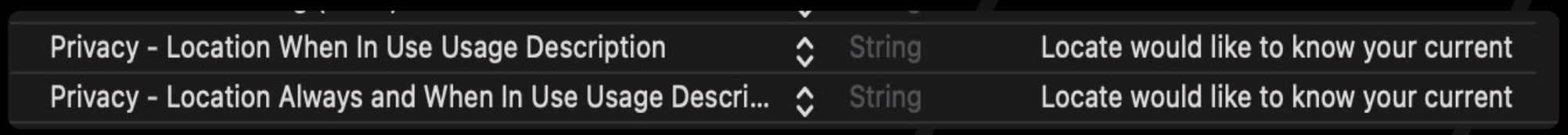
Shubham Singh
@shubham_iosdev

CORE LOCATION

1 Request permission from the user to access their location

Before you can use the location services in your app, you need to get the user's permission first.

- Add two key-value pairs in the info.plist file-
  i) Privacy- Location When In Use Usage
  ii) Privacy- Location Always and When In Use Usage



- Don't forget to add a proper description that explains why you need the permissions.

  It's shown to the user when they get the request alert.

Shubham Singh
@shubham_iosdev

2 Create a data binding class

As we will be creating a separate location manager class, to notify views about location updates we will create a binding class.

- It’s the easiest and the most commonly used way of data binding.
- It’s initialised with a value we want to observe and a function bind that does the binding and provides us the value through the Listener closure.

```
import Foundation

class BoxBind<T> {
    typealias Listener = (T) -> ()

    // MARK:- variables
    var value: T {
        didSet {
            listener?(value)
        }
    }

    var listener: Listener?

    // MARK:- initializers
    init(_ value: T) {
        self.value = value
    }

    // MARK:- functions
    func bind(listener: Listener?) {
        self.listener = listener
        listener?(value)
    }
}
```

Shubham Singh
@shubham_iosdev

3 Create a Location Manager class

Next, let's create a class that requests permission and also notifies the views on location updates.

- Import the Core Location framework & create a location manager class

- Create a static shared var, an instance of CLLocationManager.

Declare variables for properties that you need to access and put them inside a boxed variable set to nil.

```
import CoreLocation

class LocationManager: NSObject {

    // MARK:- variables
    static var shared: LocationManager = {
        return LocationManager()
    }()

    var clManager: CLLocationManager!

    var currentCity: BoxBind<String?> = BoxBind(nil)
    var currentCountry: BoxBind<String?> = BoxBind(nil)
    var currentStateCode: BoxBind<String?> = BoxBind(nil)

    // MARK:- initializers
    override init() {
        super.init()
        clManager = CLLocationManager()
        clManager.delegate = self
    }
```

Shubham Singh
@shubham_iosdev

CORE LOCATION

- Override the initializer, set the variables, and the delegate

  Create a function to request the user for their location permission.

- Create an extension of the manager class, & conform to the CLLocationManager delegate.

  The function is called whenever the app runs and when location permission changes.

```
    // MARK:- initializers
    override init() {
        super.init()
        clManager = CLLocationManager()
        clManager.delegate = self
    }

    // MARK:- functions
    func requestPermissionIfRequired() {
        clManager.desiredAccuracy = kCLLocationAccuracyBest
        clManager.requestWhenInUseAuthorization()
    }
}
```

```
extension LocationManager: CLLocationManagerDelegate {
    func locationManagerDidChangeAuthorization
                            (_ manager: CLLocationManager) {

        switch manager.authorizationStatus {

        case .notDetermined, .restricted, .denied:
            print("Location access blocked")

        case .authorizedWhenInUse, .authorizedAlways:
            print("Location permission granted")
            getLocationForUser(location: manager.location)
        @unknown default:
            break
        }
    }
```

We'll implement the getLocationForUser method next

Shubham Singh
@shubham_iosdev

CORE LOCATION

- Let’s implement the getLocationForUser function, it takes a CLLocation and we’ll use the geocoded’s reverseGecodeLocation to fetch the user’s whereabouts.

- After fetching the place mark, update the value of the boxed variables, these values can now be used by other classes.

```
private func getLocationForUser(location: CLLocation?) {
    guard let location = location else { return }

    let geocoder = CLGeocoder()
    geocoder.reverseGeocodeLocation(location, preferredLocale: .current)
    { [self] (placemarks, error) in
        if (error == nil) {
            guard let placemarks = placemarks else { return }

            let placemark = placemarks[0]
            currentCity.value = placemark.locality
            currentCountry.value = placemark.country
            currentStateCode.value = placemark.isoCountryCode
        } else {
            // handle error, notify the views through a boxed var
        }
    }
}
```

You can create one more boxed var to notify VC’s in case of an error.

Shubham Singh
@shubham_iosdev

4 Use the Location Manager

- Initialize the of LocationManager class in the AppDelegate

```
import UIKit

@main
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions
                     launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        LocationManager.shared.requestPermissionIfRequired()
        return true
    }
```

- Use the boxed variables of the Location Manager and update the UI whenever the bind function gets called.

```
// MARK:- lifeCycle for the viewController
override func viewDidLoad() {
    super.viewDidLoad()

    LocationManager.shared.currentCity.bind {
        if let currentCity = $0 {
            self.locationLabel.text = currentCity
        } else {
            self.locationLabel.text = "Unknown"
        }
    }
}
```

Screenshot from the Tutorial App

Shubham Singh
@shubham_iosdev

CORE LOCATION

RECAP

1. Request permission from the user to access their location
2. Create a data binding class to update UI
3. Create a Location Manager class where you implement the Core Location APIs.
4. Initialize the Location Manager in the App Delegate
5. Use the bind function of the boxed variables in the ViewController / Views.

Shubham Singh
@shubham_iosdev

CORE LOCATION

Shubham Singh
@shubham_iosdev

We've only implemented a single function so far. You can go ahead and try out the other functions of the Core Location Framework.

What would you like to learn next?

LIKE

SAVE FOR LATER