BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

MAD6

Ortung

Prof. Dr. Dragan Macos

CoreLocation

- Das ist ein Framework für die Ortsbestimmung auf den Planeten. Am besten auf der Erde.
- Basierend auf dem Global Positioning System(GPS)
- Seit Mitte 1990 voll funktionsfähig.
- Insgesamt 8 Satelliten kreisen um die Erde.
- Sie broadcasten einfach Ihre Position und die Sendezeit.
- Auf Grund der Position und der Sendezeit können wir zum Zeitpunkt des Empfangs die Entfernung vom sendenden Satelliten herleiten.
- Bei einem Satelliten: Kugelförmige Entfernung, 2 Satelliten: Kreis, bei 3 Satelliten 2 Punkte, wobei einer imaginär ist. Also 3 Satelliten reichen für die Ortsbestimmung selbst.
- Für die GPS-Ortsbestimmung werden 4 Satelliten benötigt
- 3 Satelliten für die eigentliche Ortsbestimmung und einer für die Uhrkorrektur
- Eine Sekunde Ungenauigkeit bei der Uhr -> großer Ortungsfehler

CoreLocation

- CoreLocation greift das GPS-Signal vom GPS-Modul des Gerätes ab (iPhone, iPad), wenn das Modul im Gerät vorhanden ist.
- Unserer Applikation stellt er viele schicken Funktionen zur Verfügung.
- Wir werden hier nur die wichtigsten erwähnen und das Verfahren für die Integration von CoreLocation in unsere Applikation klären. Alle Funktionen von Core Location sind in dessen API-Spezifikation zu finden.
- ... Und jetzt geht‘s los....

Die GPS-Koordinaten

- Der grundlegende Typ des CoreLocation-Frameworks: die Klasse CLLocation

Location Attributes

`coordinate`

The geographical coordinate information. (read-only)

Declaration

SWIFT

```
var coordinate: CLLocationCoordinate2D { get }
```

Coordinate

Coordinate hat zwei Properties:

- latitude
  → Geographische Breite

- longitude
  → Geographische Länge

CoreLocation importieren

Wir müssen dem System sagen, dass wir CoreLocation verwenden möchten:

```
import CoreLocation
```

Ähnlich, wie in Java. Die Methoden/Properties des Frameworks sind nun in unserem Programm aufrufbar.

Wir schauen uns das Framework CoreLocation an.

→ Apple developer Dokumentation.

CoreLocation Framework Reference

Classes Protocols Other Reference

3 Wichtige Bestandteile der Dokumentation

The Core Location framework lets you determine the current location or h…amework uses the available hardware to determine the user's position and heading. You use the clas…nfigure and schedule the delivery of location and heading events. You can also use it to define geographic regions and monitor when the user crosses the boundaries of those regions. In iOS, you can also define a region around a Bluetooth beacon.

Classes

Das gilt für swift nicht.

Class	Abstract
NSObject	NSObject is the root class of most Objective-C class hierarchies.
CLBeacon	The CLBeacon class represents a beacon that was encountered during region monitoring.
CLFloor	A CLFloor object specifies the floor of the building on which the user is located.
CLGeocoder	The CLGeocoder class provides services for converting between a coordinate (specified as a latitude and longitude) and the user-friendly representation of that coordinate.
CLHeading	A CLHeading object contains heading data generated by a CLLocationManager object.
CLLocation	A CLLocation object represents the location data generated by a CLLocationManager object.
CLLocationManager	The CLLocationManager class is the central point for configuring the delivery of location- and heading-related events to your app.
CLPlacemark	A CLPlacemark object stores placemark data for a given latitude and longitude.
CLRegion	The CLRegion class defines an abstract area that can be tracked.
CLBeaconRegion	A CLBeaconRegion object defines a type of region that is based on the device's proximity to a Bluetooth beacon, as opposed to a geographic location.
CLCircularRegion	The CLCircularRegion class defines the location and boundaries for a circular geographic region.
CLVisit	A CLVisit object encapsulates information about interesting places that the user has been.

CLLocationManager

Accessing the Delegate

`delegate`

Initiating Standard Location Updates

`startUpdatingLocation()`

`stopUpdatingLocation()`

`pausesLocationUpdatesAutomatically`

`distanceFilter`

`desiredAccuracy`

`activityType`

```swift
var desiredAccuracy: CLLocationAccuracy
```

```swift
func requestAlwaysAuthorization()
```

`delegate`

The delegate object to receive update events.

Declaration

```swift
unowned(unsafe) var delegate: CLLocationManagerDelegate!
```

CLLocationManager: Der Ortungsboss

Accessing the Delegate

`delegate`

Initiating Standard Location Updates

`startUpdatingLocation()`

`stopUpdatingLocation()`

`pausesLocationUpdatesAutomatically`

`distanceFilter`

`desiredAccuracy`

`activityType`

Wenn keiner mehr auf delegate mit „Strong“ zeigt, dann wird das delegate deallociert.. Darüber später. Kleine Show an der Tafel

Melde keinen Fehler, wenn dieser Pointer nach Deallozierung referenziert wird.
Dagegen wird bei „safe“ ein Laufzeitfehler bei der Refferenzierung gemeldet.
DARÜBER SPÄTER MEHR ☺

`delegate`

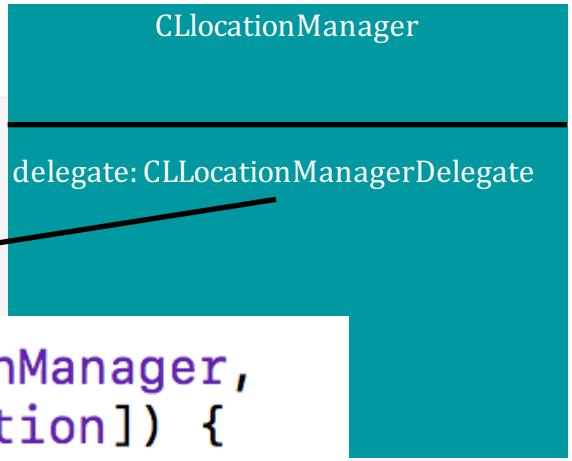
The delegate object to receive update events.

Declaration

SWIFT

```
unowned(unsafe) var delegate: CLLocationManagerDelegate!
```

Das Protokoll CLLocationManagerDelegate vom delegate des CLLocationManager



Tells the delegate that new location data is available.

Declaration

eine Methode des Protokolls

```
func locationManager(_ manager: CLLocationManager,
    didUpdateLocations locations: [CLLocation]) {
```

Diese Methode wird aufgerufen, wenn die Positionsdaten sich geändert haben!!

CLLocationManager

`location`

The most recently retrieved user location. (

Declaration

```
SWIFT
@NSCopying var location: CLLocation! { get }
```

Alternative zu locations[]

locationManager
delegate: CLLocationManagerDelegate var location: CLLocation! {get}

Instance Method

locationManager(_:didUpdateLocations:)

Tells the delegate that new location data is available.

locationManager

delegate: CLLocationManagerDelegate

var location: CLLocation! {get}

Declaration

```
func locationManager(_ manager: CLLocationManager,
    didUpdateLocations locations: [CLLocation])
```

Wer ist delegate?

Parameters

`manager`
The location manager object that generated the update event.

`locations`
An array of `CLLocation` objects containing the location data. This array always contains at least one object representing the current location. If updates were deferred or if multiple locations arrived before they could be delivered, the array may contain additional entries. The objects in the array are organized in the order in which they occurred. Therefore, the most recent location update is at the end of the array.

Wir möchten für den Anfang nur die GPS-Koordinaten ausgeben..

Was machen wir jetzt damit??

DISKUSSION!

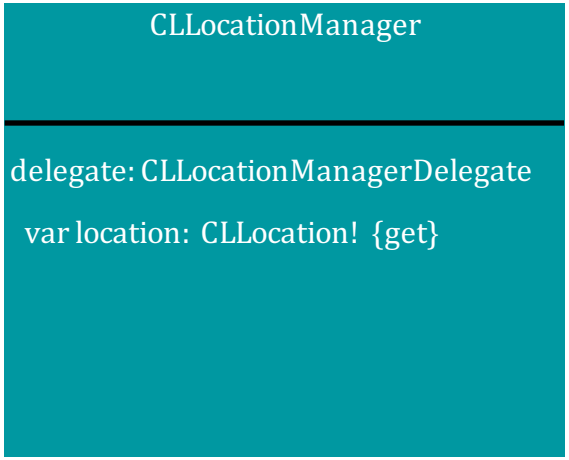
Schritt 1

```
import CoreLocation

class ViewController: UIViewController, CLLocationManagerDelegate{
```

CLLocationManager
delegate: CLLocationManagerDelegate var location: CLLocation! {get}

Schritt 2

```
var locationManager = CLLocationManager()
locationManager.delegate = self
```



Schritt 3

- Die Genauigkeit setzten

```
locationManager.desiredAccuracy = kCLLocationAccuracyBest
```

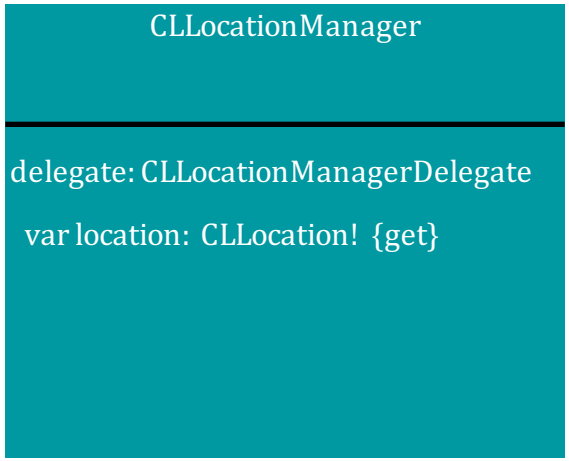
- Gesetzlich bestimmt oder Apple-Anforderung: Immer fragen, ob das Gerät die Position tracken darf.

```
locationManager.requestAlwaysAuthorization()
```

Schritt 4

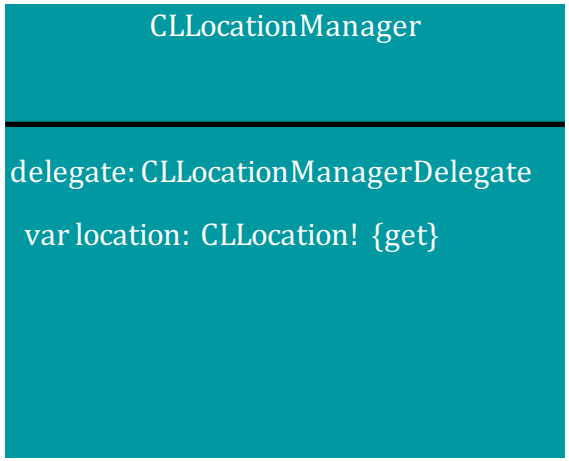
```
locationManager.startUpdatingLocation()
```

- Jetzt wird der locationManager immer benachrichtigt, wenn die Positionsdaten sich geändert haben.



Schritt 5

- Was jetzt??



Schritt 5

- Methode aus dem CLLocationManagerDelegate-Protokoll umsetzen.

```
func locationManager(_ manager: CLLocationManager,
    didUpdateLocations locations: [CLLocation]) {
```

Zwei Möglichkeiten, die Position rauszubekommen

Declaration

SWIFT

```
@NSCopying var location: CLLocation! { get }
```

Property des CLLocationManager

Schritt 5

```
var locationManager = CLLocationManager()
locationManager.delegate = self
```

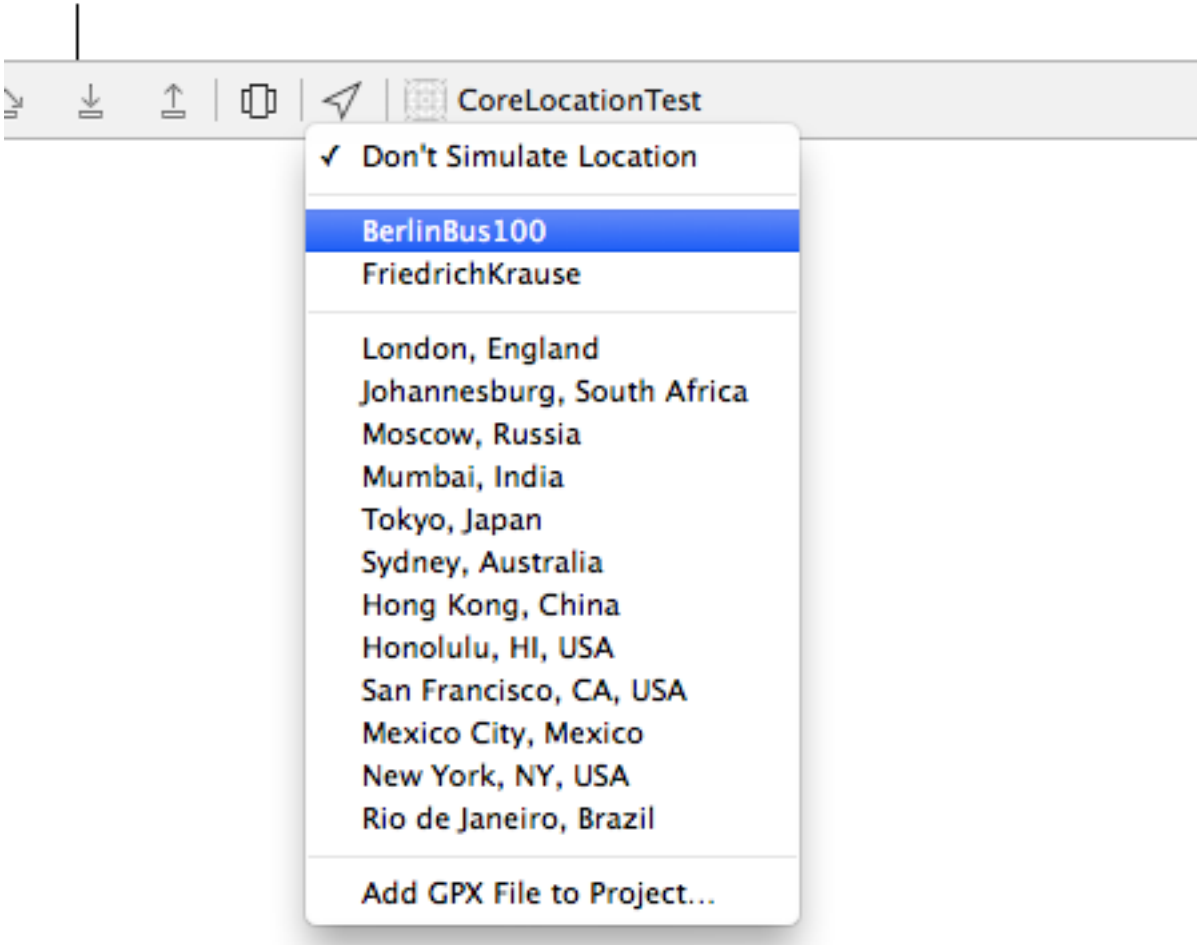
- Methode aus dem CLLocationManagerDelegate-Protokoll umsetzen.

```
func locationManager(_ manager: CLLocationManager,
    didUpdateLocations locations: [CLLocation]) {
    let loc = locations.last
    let latVal = loc?.coordinate.latitude
    let lonVal = loc?.coordinate.longitude
    print("LAT:"+(latVal?.description)!)
    print("LON:"+(lonVal?.description)!)
    print()

    print("**** from locations:")
    print("LAT:"+
        (manager.location?.coordinate.latitude.description)!)
    print("LON:"+(manager.location?.coordinate.longitude.
        description)!)
    print()


}
```

Test

- Wir haben spezielle Tragetaschen für Mac-Minis entwickelt.
- Sie können Ihren Mac-Mini in die Tasche tun und eine Runde auf dem BHT-Gelände laufen.
- Die GPS-Koordinaten werden als Log gespeichert und Sie schauen sich wieder im Labor die Ergebnisse an.

GPS-Tracks auswählen

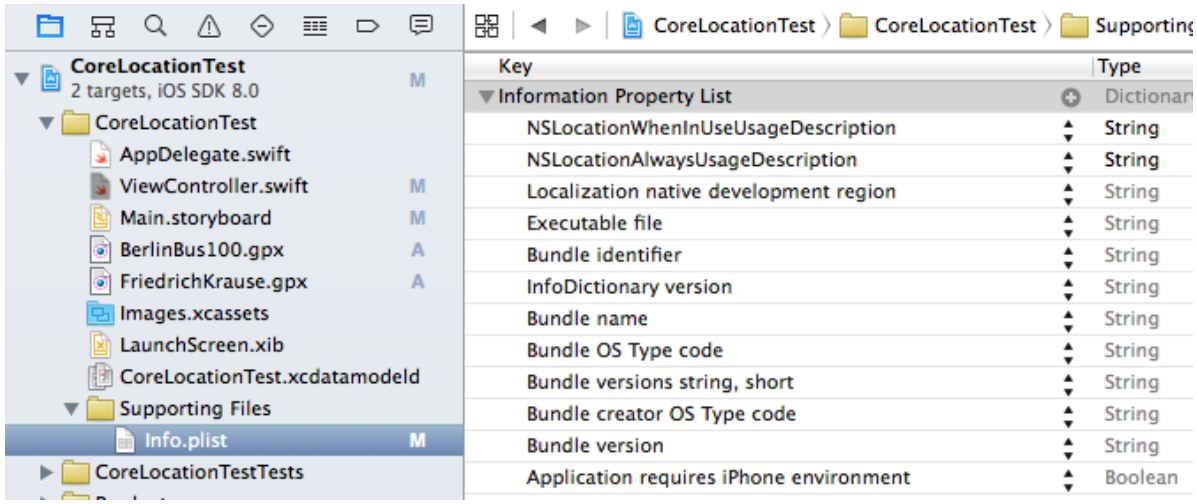


Ausgabe

```
**** from locations:
LAT:52.505414    LON:13.332905
**** from locationManager:
LAT:52.505414    LON:13.332905
**** from locations:
LAT:52.505235    LON:13.333229
**** from locationManager:
LAT:52.505235    LON:13.333229
**** from locations:
LAT:52.505088    LON:13.333574
**** from locationManager:
```

Noch eine Sache

- Wichtige Einstellung im xCode:



- 3 Parameter eintragen (Sicherheit) ☺
  - NSLocationWhenInUseUsageDescription
  - NSLocationAlwaysUsageDescription
  - NSLocationAlwaysAndWhenInUseUsageDescription

Quellen

Die meisten Sourcecode-Beispiele und die Sprachdefinition der Sprache Swift wurden aus:

Apple Inc. „The Swift Programming Language.“ iBooks. https://itun.es/de/jEUH0.l

genommen.

Eventuelle andere Quellen bzw. eigene Beispiele werden an den entsprechenden Stellen direkt angegeben bzw. gekennzeichnet.