iOS DeCal : Lecture 5

Core Location, MapKit, AVFoundation, and Core Data

March 7, 2017

Announcements

Lab 4 (Pokedex) now due next Tuesday (11:59pm)

No new lab will be assigned this week

This week lab : Lab 4 + Project 2 help

Attendance still required

Project 2 Part 1(Snapchat Clone) due next Tuesday

Note on Gradescope Submissions

Please re-download your submission to make sure it works locally

Images only get preserved when you use the Github submission feature.

Make sure all the extra files you use are copied into the project directory

If you received a low score due to an application bug, either resubmit or come show us your app after class

Overview : Today's Lecture

Core Location

Map Kit

AVFoundation

Core Data

Core Location

Review: : Internet - XPS

GPS drains battery and is unreliable in dense urban and indoor environments

Need accurate location services…

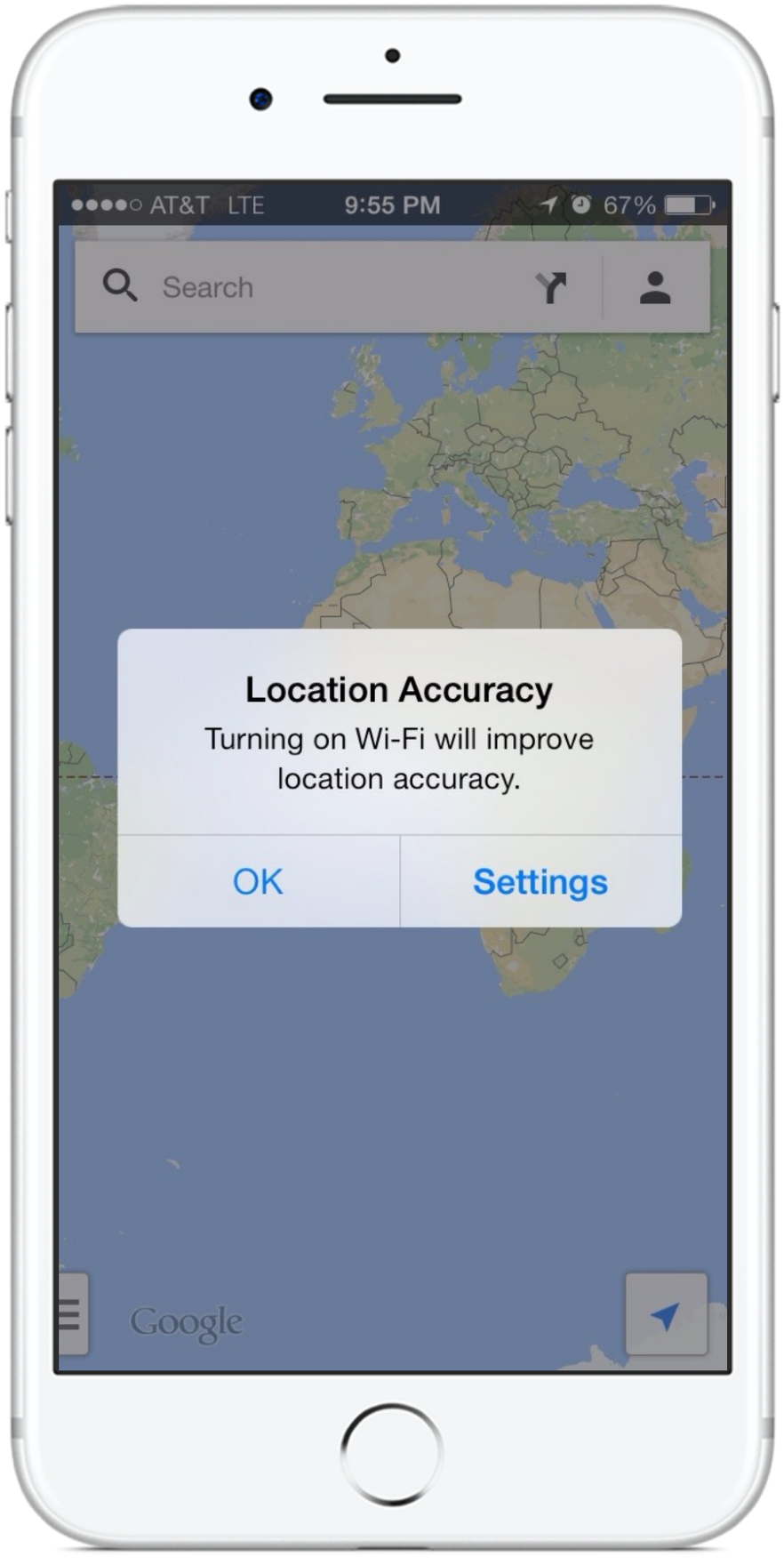
What can you do?

Review: : Internet - XPS

XPS - Hybrid Positioning System

Use crowd-sourced database of Wi-Fi hotspot and cell tower locations

Core Location



Core Location : Permissions

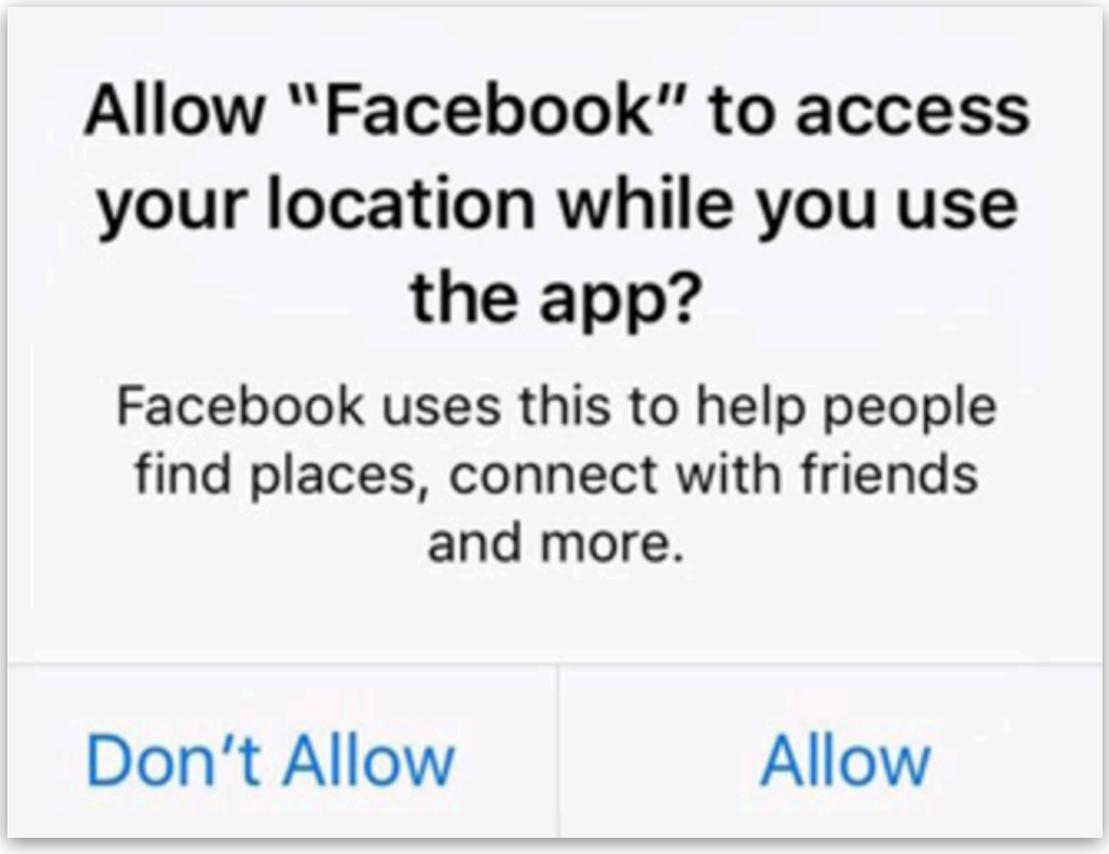
Before getting a user's location, they have to have enabled Location Services for your app

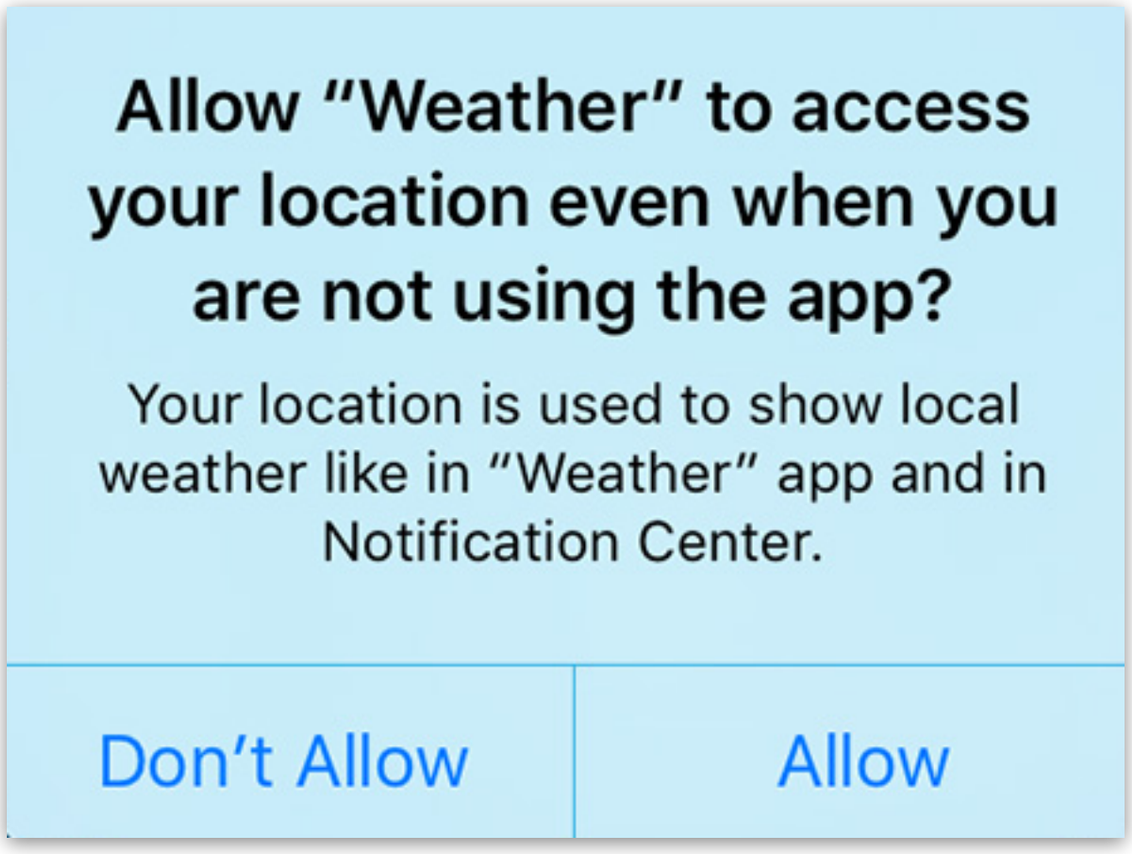
```
let manager = CLLocationManager()

if !CLLocationManager.locationServicesEnabled()
{
    //ask for user's location
}
```

Core Location : Permissions

When In Use vs. Always






```
manager.requestWhenInUseAuthorization()

manager.requestAlwaysAuthorization()
```

Core Location : Permissions

Let’s say you always want the user’s location

Even when not in the app (background)

```
switch CLLocationManager.authorizationStatus() {
    case .authorizedAlways:
        break
     case .notDetermined:
        manager.requestAlwaysAuthorization()
     case .authorizedWhenInUse, .restricted, .denied:
        //prompt notification: see next slides
}
```

Alerts

```
let alertController = UIAlertController(
        title: "Background Location Access Disabled",
        message: "In order to ____, please open
            Settings and set location access
            for this app to 'Always'.",
    preferredStyle: .alert)
```

Alerts

```
let openAction = UIAlertAction(title: "Open Settings",
                               style: .default) { (action) in
    if let url = NSURL(string: UIApplicationOpenSettingsURLString) {
        UIApplication.shared.open(url as URL,
                                  options: [:],
                                  completionHandler: nil)
    }
}

alertController.addAction(openAction)
```

Alerts

```
let cancelAction = UIAlertAction(title: "Cancel",
                                 style: .cancel,
                                 handler: nil)
```

```
alertController.addAction(cancelAction)
```

Alerts

To actually present the alert in your desired context...

```
self.present(alertController,
             animated: true,
             completion: nil)
```

Core Location : One Time Location

Fetch user's location once

```
let manager = CLLocationManager()

override func viewDidLoad() {
    super.viewDidLoad()
    // manager is your CLLocationManager
    manager.delegate = self //important!!
    manager.desiredAccuracy =
    kCLLocationAccuracyBest
    manager.requestLocation() //type of update
}
```

Core Location : One Time Location

Calling the method

```
manager.requestLocation()
```

Will call either:

```
locationManager(_:didUpdateLocations:)
locationManager(_:didFailWithError:)
```

from your `CLLocationManagerDelegate` class (that's why you must set the delegate!)

Core Location : Location over Time

Standard Location Service

For continuous updates (e.g. Maps)

```
manager.startUpdatingLocation()
```

Significant-Change Loc. Service

Update only when location changes

```
manager.startMonitoringSignificantLocationChanges()
```

Core Location : Location over Time

Must implement appropriate delegate method(s) in View Controller to receive data

```
func locationManager(manager: CLLocationManager,
                     didUpdateLocations locations:
                                          [CLLocation]) {
    // most recent location update at the end of the array
    let latestLocation = locations[locations.count - 1]
    // do something
}
```

Core Location : CLVisit

A period of time a user has spent in a single location, including both a coordinate and start/end timestamps

```
// initiating visit event updates
manager.startMonitoringVisits()
manager.stopMonitoringVisits()

// receive data via delegate method(s)
func locationManager(manager: CLLocationManager,
                     didVisit visit: CLVisit) {
}
```

Core Location : CLRegion

Monitor boundary crossings for defined geographical regions (geofencing)

```
// define desired geographical region and radius
let currRegion = CLCircularRegion(center:
          CLLocationCoordinate2D(latitude: 37,
                                 longitude: 122),
                                 radius: 200,
                                 identifier: "Berkeley")

// initiating region monitoring
manager.startMonitoring(for: currRegion)
manager.stopMonitoring(for: currRegion)
```

Core Location : CLRegion

Monitor boundary crossings for defined geographical regions (geofencing)

```
//delegate method fires when user enters
func locationManager(manager: CLLocationManager,
didEnterRegion region: CLRegion) { … }


//delegate method fires when user exits
func locationManager(manager: CLLocationManager,
didExitRegion region: CLRegion) { … }
```

Core Location : Other

CLFloor - get information about what floor your user is on (returns int for floor)

iBeacons: developer.apple.com/ibeacon/

And more...

Core Location : User Trust

Keep location data secure

Do not auto-track user

Only use Location Services when they are needed

Map Kit

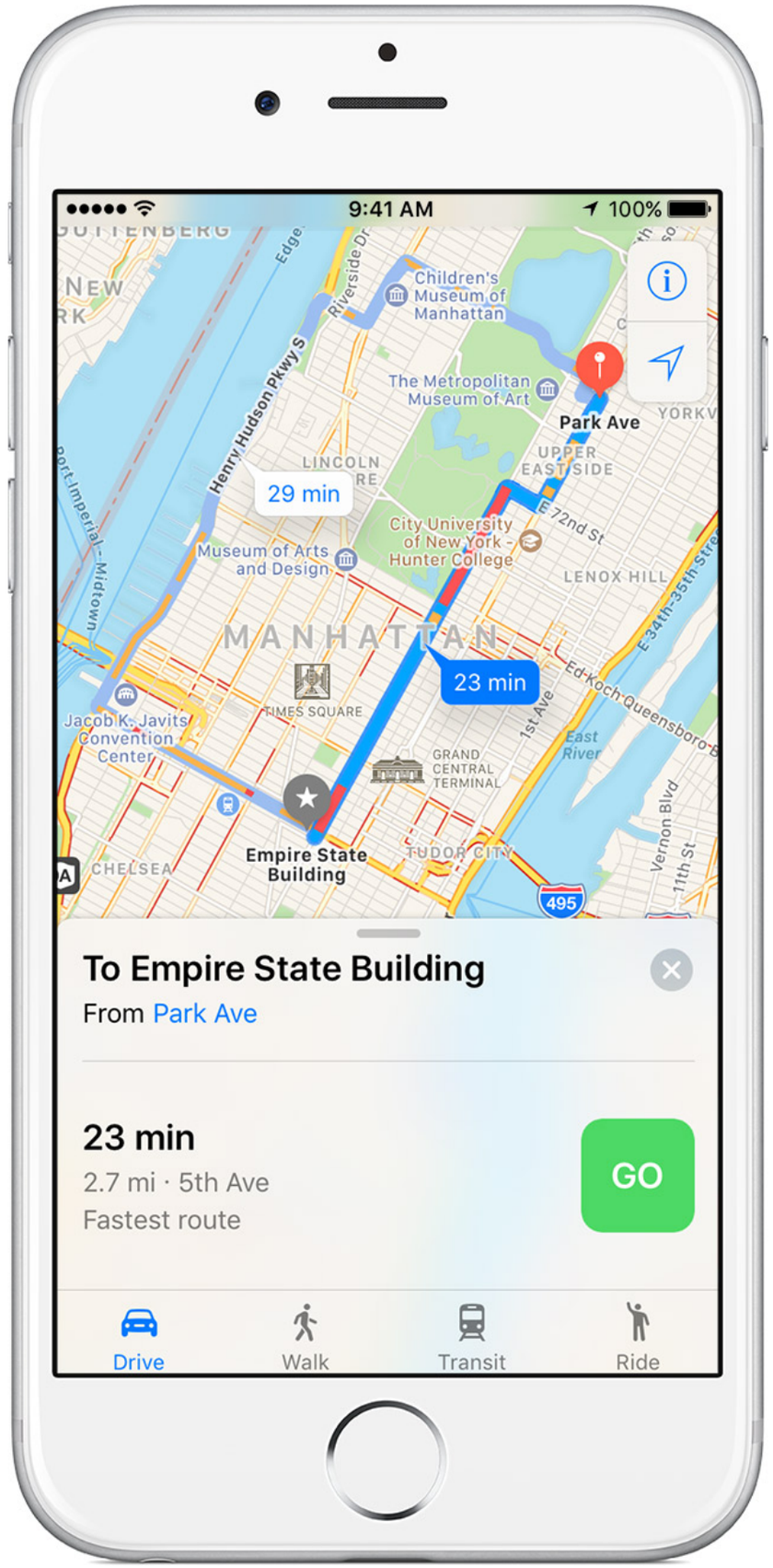
MapKit - Overview

API built off of CoreLocation

Embed maps directly to windows or views

Some Features:

- Annotate Map & Add Overlays
- Plot Location
- Jump to coordinates



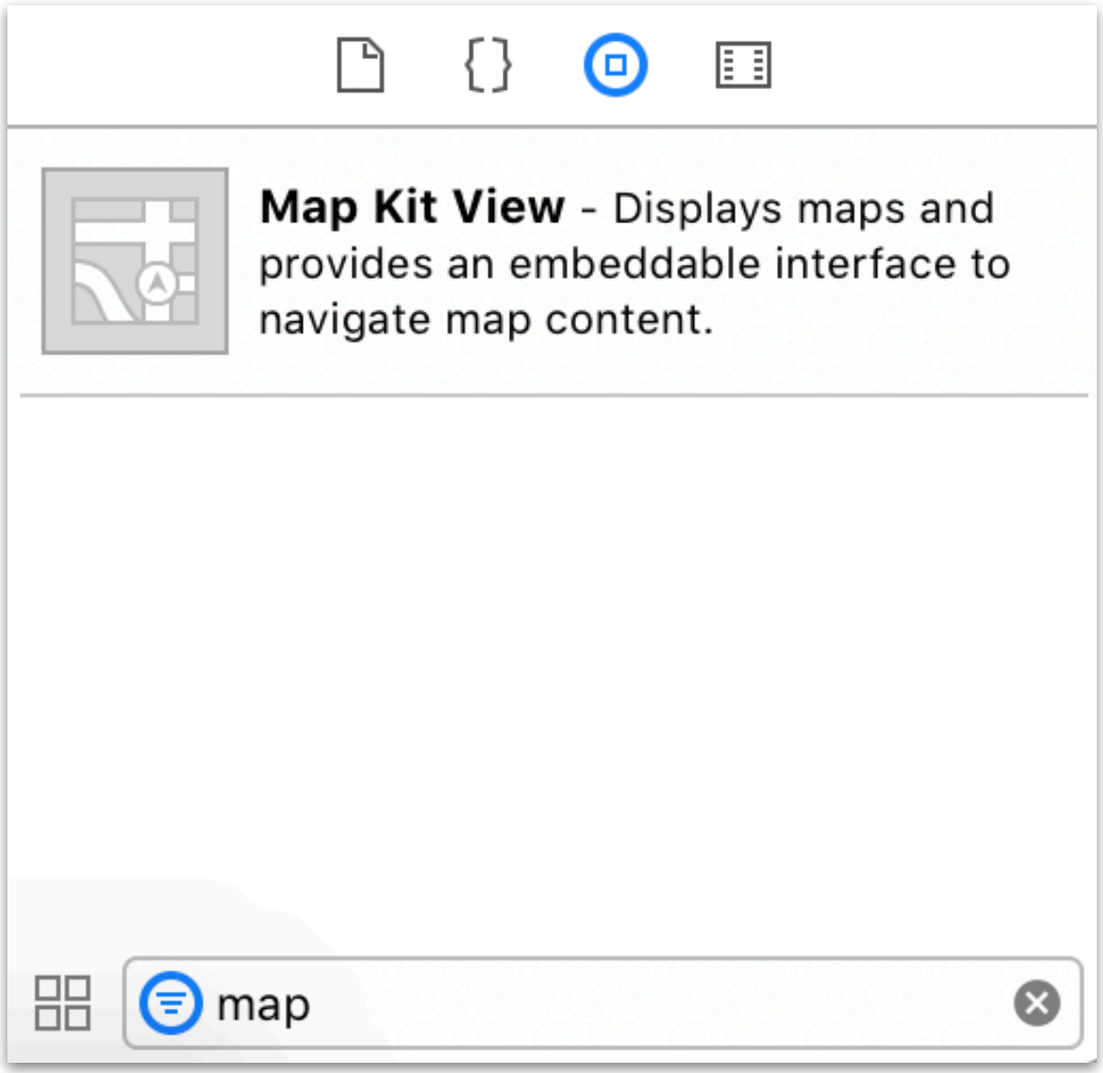
MapKit Example

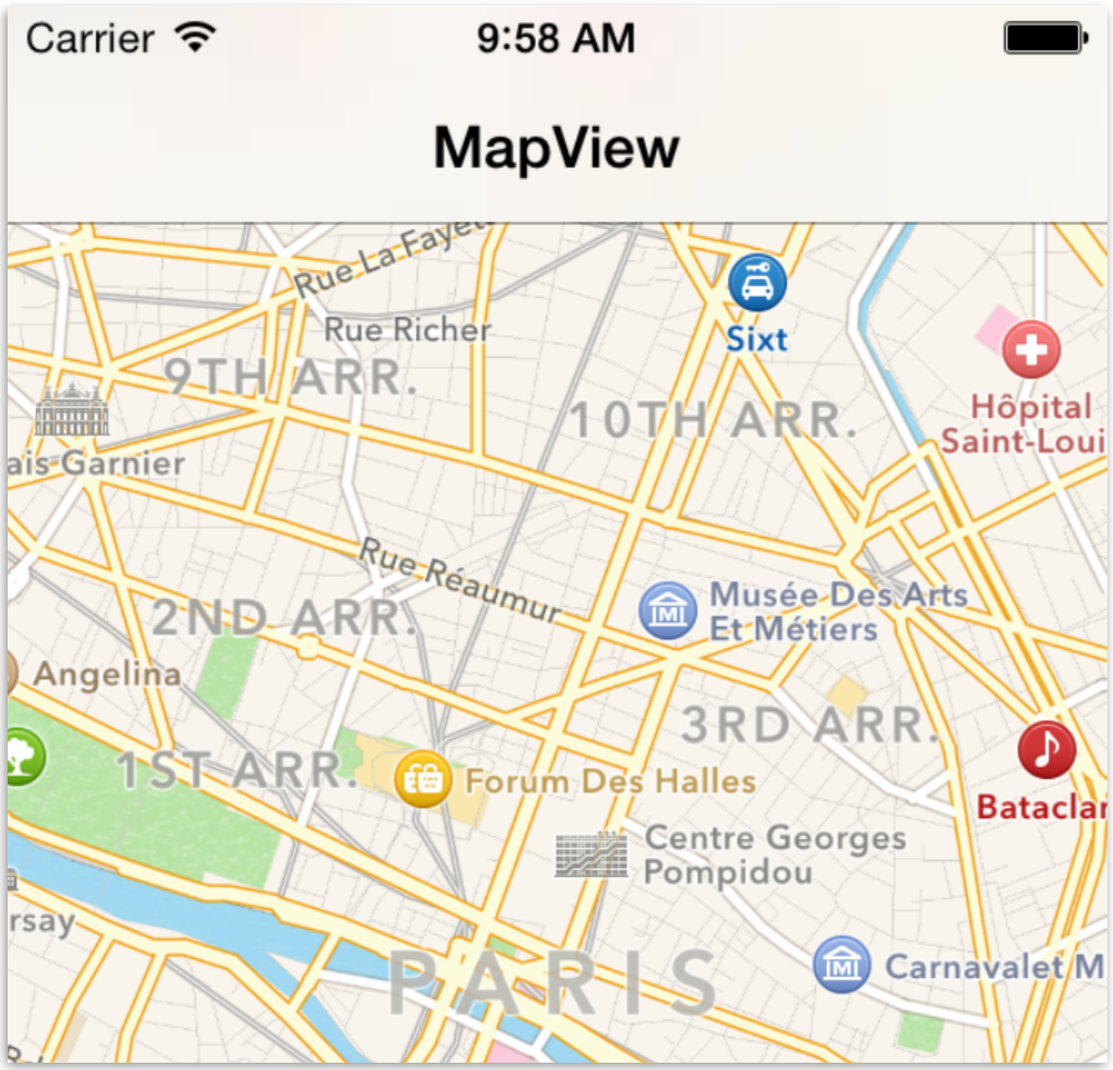
Easily embed an interactive map within your application with annotations

Example: Yelp

Embedding a Map : Storyboard

Drag a “Map Kit View” from the Object Library into your View Controller.





Custom Initial Map View

```
override func viewDidLoad() {
    super.viewDidLoad()
    let initialLoc = CLLocation(latitude:38,
                                longitude:122)
    centerMapOnLocation(initialLoc) //custom
}
```

Custom Initial Map View

```
// in your map's view controller
let regionRadius: CLLocationDistance = 1000

func centerMapOnLocation(location: CLLocation) {
    let coordinateRegion =
        MKCoordinateRegionMakeWithDistance(
            location.coordinate,
            regionRadius * 2.0,
            regionRadius * 2.0)
    mapView.setRegion(coordinateRegion,
                      animated: true)
}
```

Adding Annotations

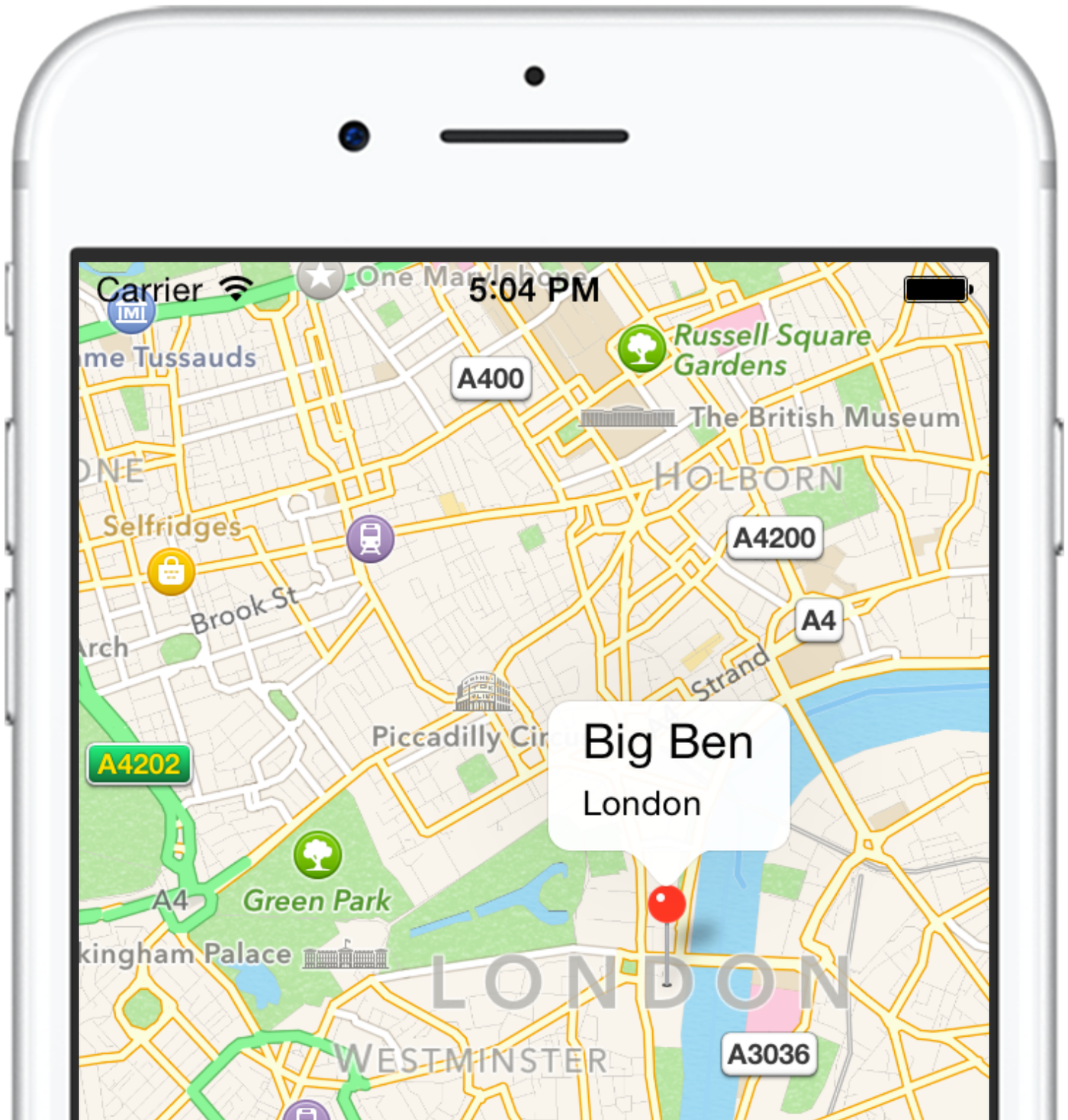
```
override func viewDidLoad() {
    let annotation = MKPointAnnotation()
    annotation.coordinate =
            CLLocationCoordinate2D(
                    latitude: 24,
                    longitude: 54)
    annotation.title = "Big Ben"
    annotation.subtitle = "London"
    mapView.addAnnotation(annotation)
}
```

What we have done so far

Created a Initial Map View

Set a location

Added an **Annotation** (with a title, subtitle, and coordinated)



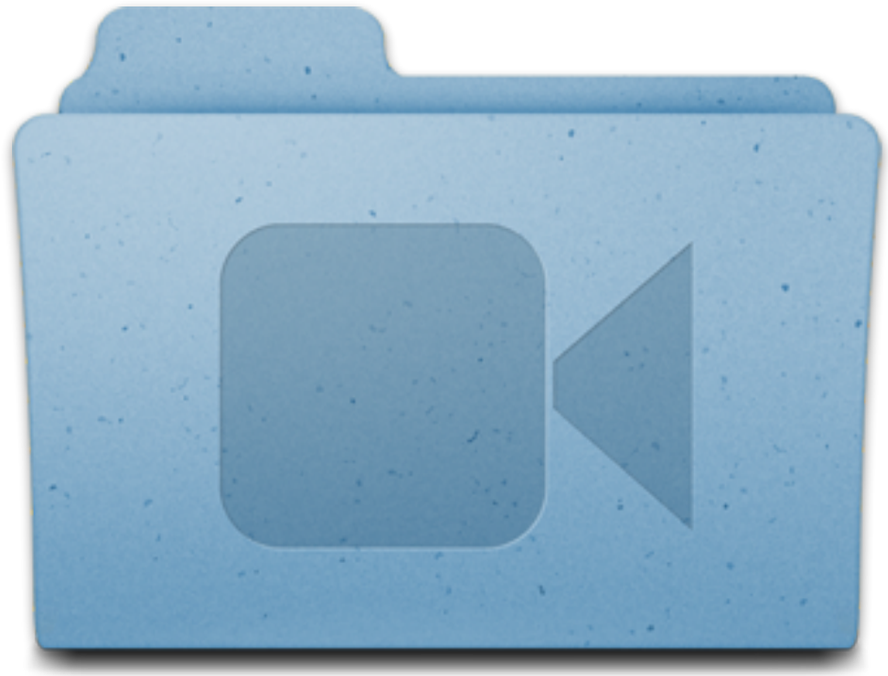
Check In

AVFoundation

AVFoundation - What is it?

Cocoa framework for
AudioVisual items

- Used to record, edit, and stream media
- Includes Players, Items, ViewControllers, etc



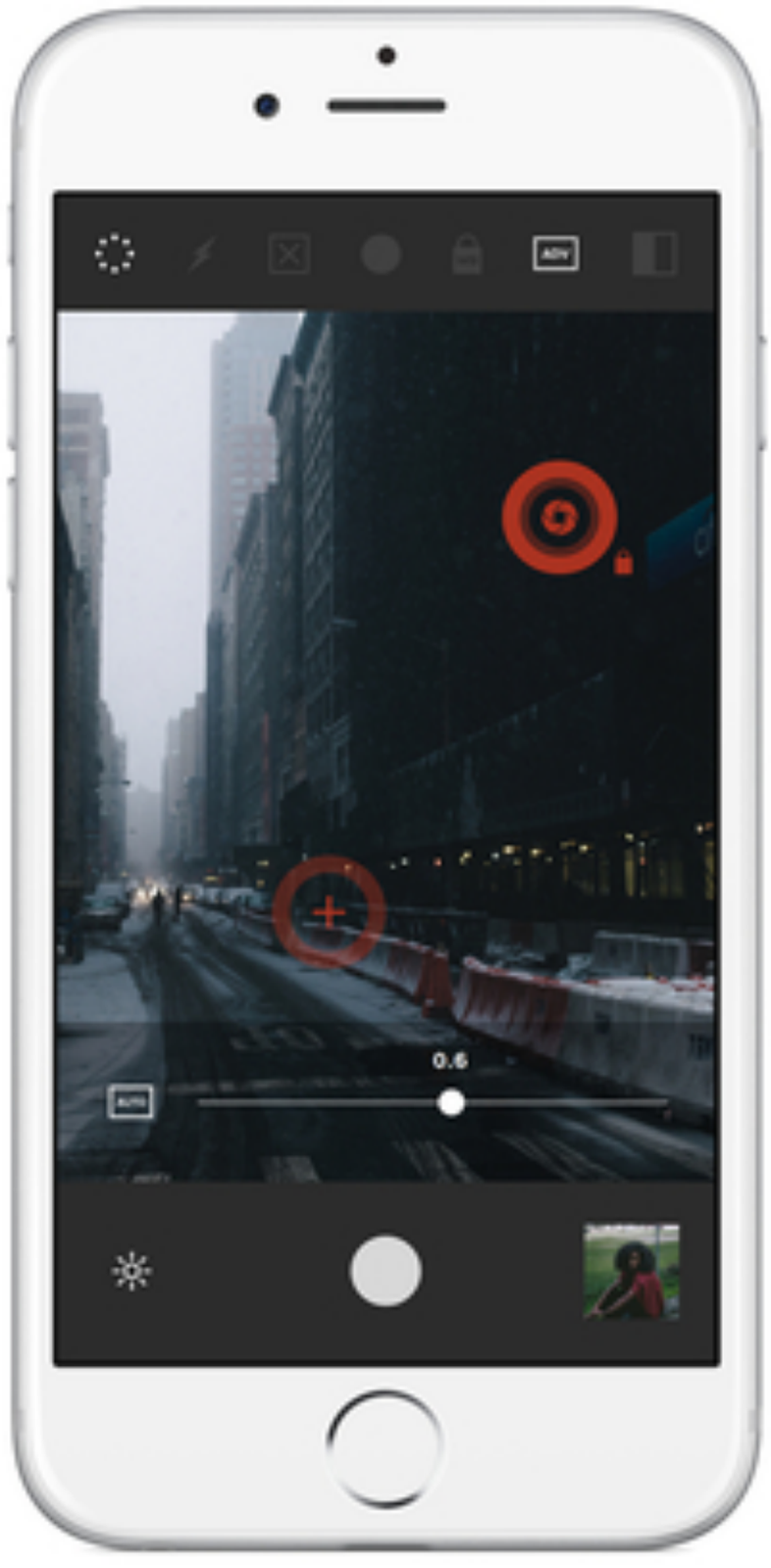
AVCapture - Overview

Allows you to capture video, photo, scan barcodes, etc.

Create a **AVCaptureSession**

Set the **AVCaptureDeviceInput** depending on what you want to capture (video, photo)

Begin media capture by calling `startRunning()` on your session

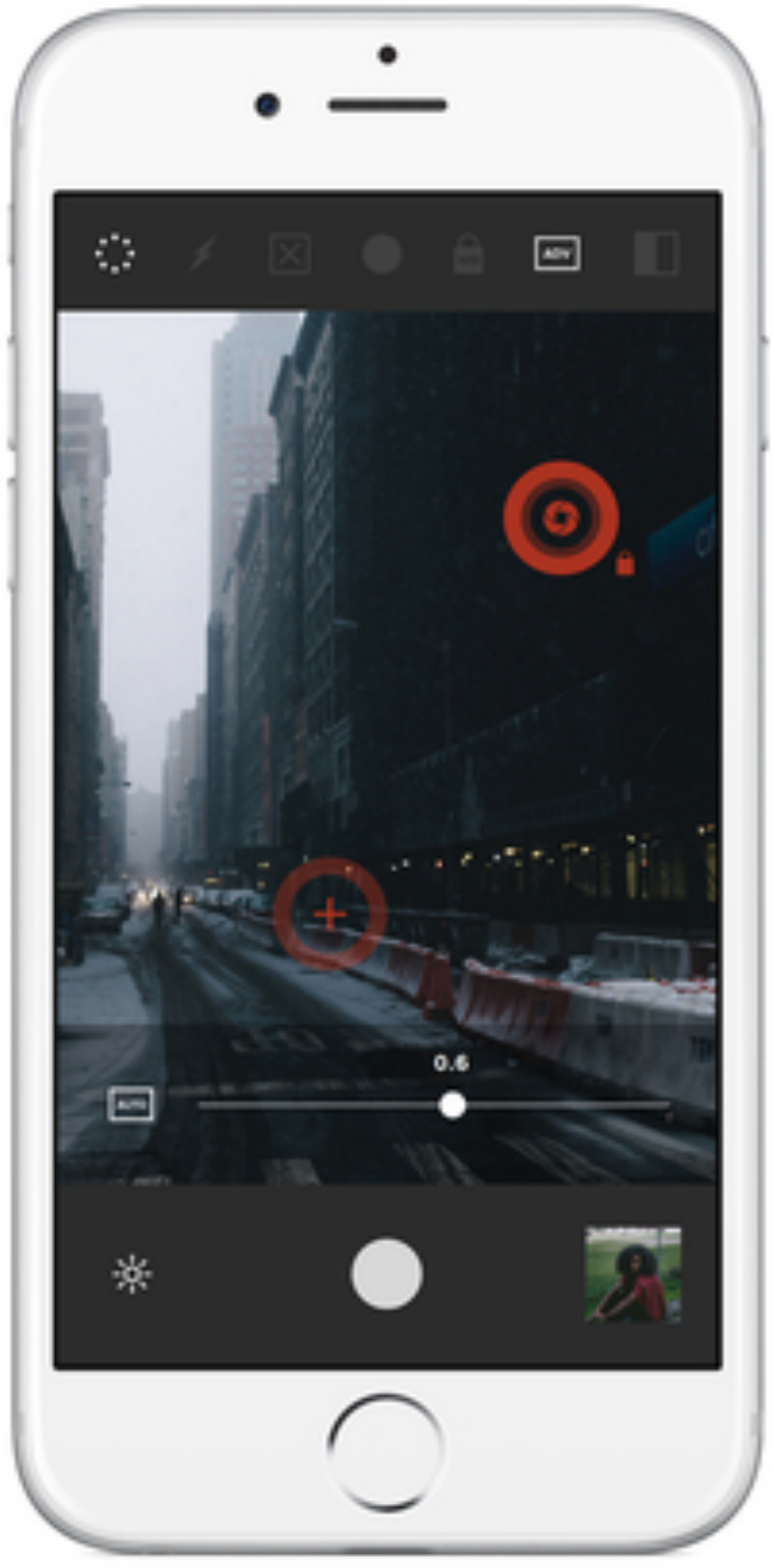


VSCO

AVCapture - Overview

Allows you to capture video, photo, scan barcodes, etc.

Note: If you just need to capture photo and video without custom formatting, use the UIKit framework instead (check out `UIImagePickerController`)



VSCO

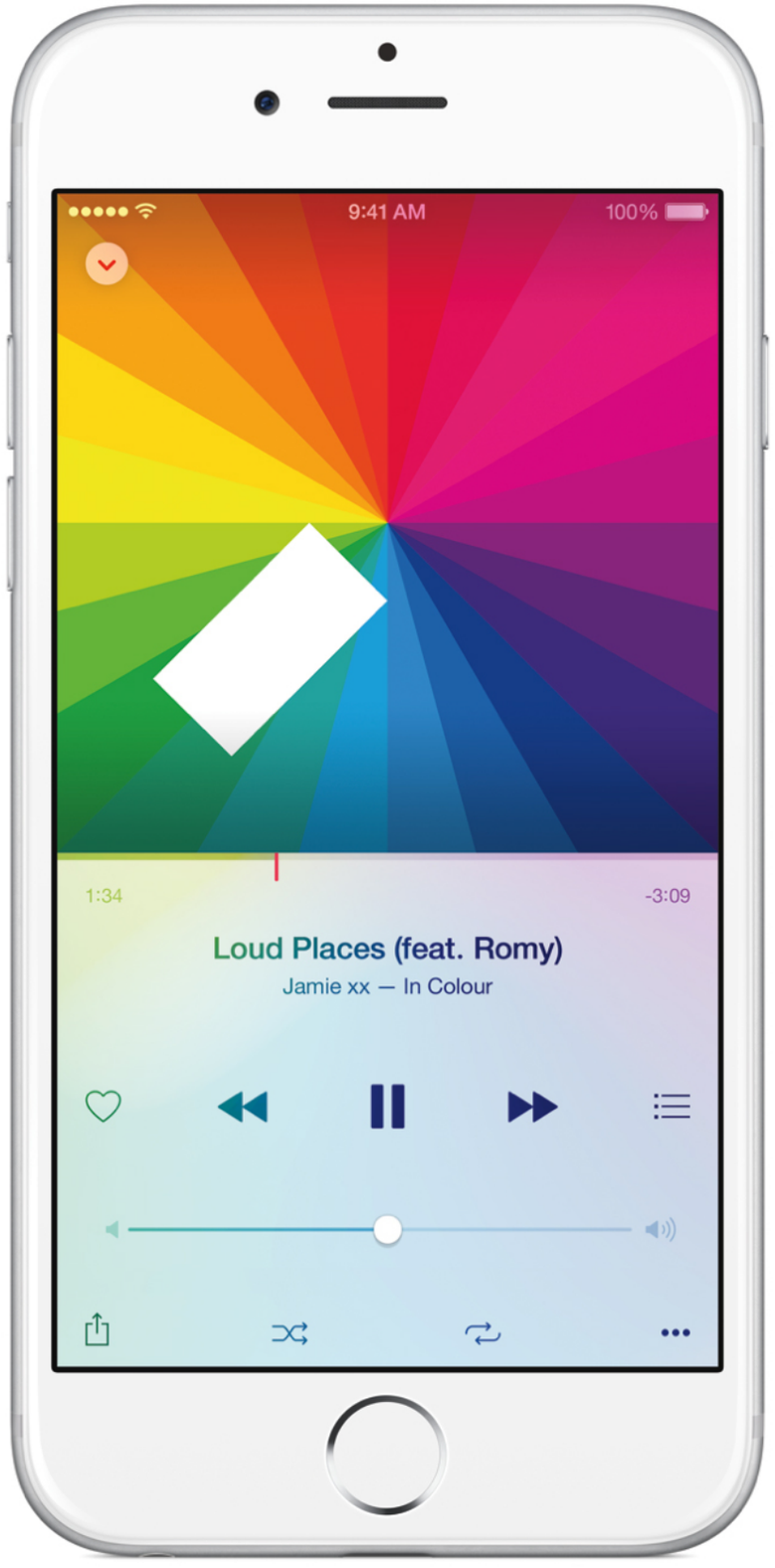
AVPlayer - Overview

Play audio in your App

Create an **AVAudioPlayer** to play your audio on

Create an **AVPlayerItem** for each sound clip / song

Use the player to play, pause, rewind, and fast forward your **AVPlayerItems**



Apple Music

AVPlayerItem - Initialization

Create an **AVPlayerItem** for each song / sound you want played.

Each **AVPlayerItem** is a single instance being played by AVPlayer

```
let item = AVPlayerItem(URL: someURL!)
```

AVPlayer - Initialization

Initialize an **AVPlayer** with or without a Player Item. You will add items to the player, then use the player to play these items.

```
let player = AVPlayer()

let player = AVPlayer(playerItem: item)
```

AVPlayer - Playback

Once you've added some AVPlayerItems to your player, you can play, pause, fast forward, replace, etc.

```
let player = AVPlayer(playerItem: item)

player.play()
player.pause()
player.seek(to: <CMTime>)
player.replaceCurrentItem(with: newSong)
```

AVPlayer - Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
    let song = AVPlayerItem(url: songURL)
    let player = AVPlayer(playerItem: song)
    if (player.currentItem!.status == .readyToPlay) {
        player.play()
    }
}
```

AVPlayer - Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
    let song = AVPlayerItem(url: songURL)
    let player = AVPlayer(playerItem: song)
    if (player.currentItem!.status == .readyToPlay) {
        player.play()
    }
}
```

Import the **AVFoundation framework**
at the top of your file

AVPlayer - Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
    let song = AVPlayerItem(url: songURL)
    let player = AVPlayer(playerItem: song)
    if (player.currentItem!.status == .readyToPlay) {
        player.play()
    }
}
```

Create an **AVPlayerItem** from a url or file in your application

AVPlayer - Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
    let song = AVPlayerItem(url: songURL)
    let player = AVPlayer(playerItem: song)
    if (player.currentItem!.status == .readyToPlay) {
        player.play()
    }
}
```

Add that **AVPlayerItem** to an **AVPlayer** (here, we are initializing the **AVPlayer** with the item)

AVPlayer - Example

```
import AVFoundation
...

func playSongFromURL(songURL: URL) {
    let song = AVPlayerItem(url: songURL)
    let player = AVPlayer(playerItem: song)
    if (player.currentItem!.status == .readyToPlay) {
        player.play()
    }
}
```

Now you can play, pause, seek, etc.

Core Data

Core Data : What is it?

Framework that allows you to store and retrieve data from a database in an OOP way

Allows **data persistence**

This lets users store data in your application, that will persist between application launches

Use it to create **data models** that can be added to and queried throughout your project

Core Data : What is it?

When do you want to use Core Data?

Examples

Allow users to save specific podcasts to their phone

Notes application that stores text + photos to your phone

Any app requiring saving state

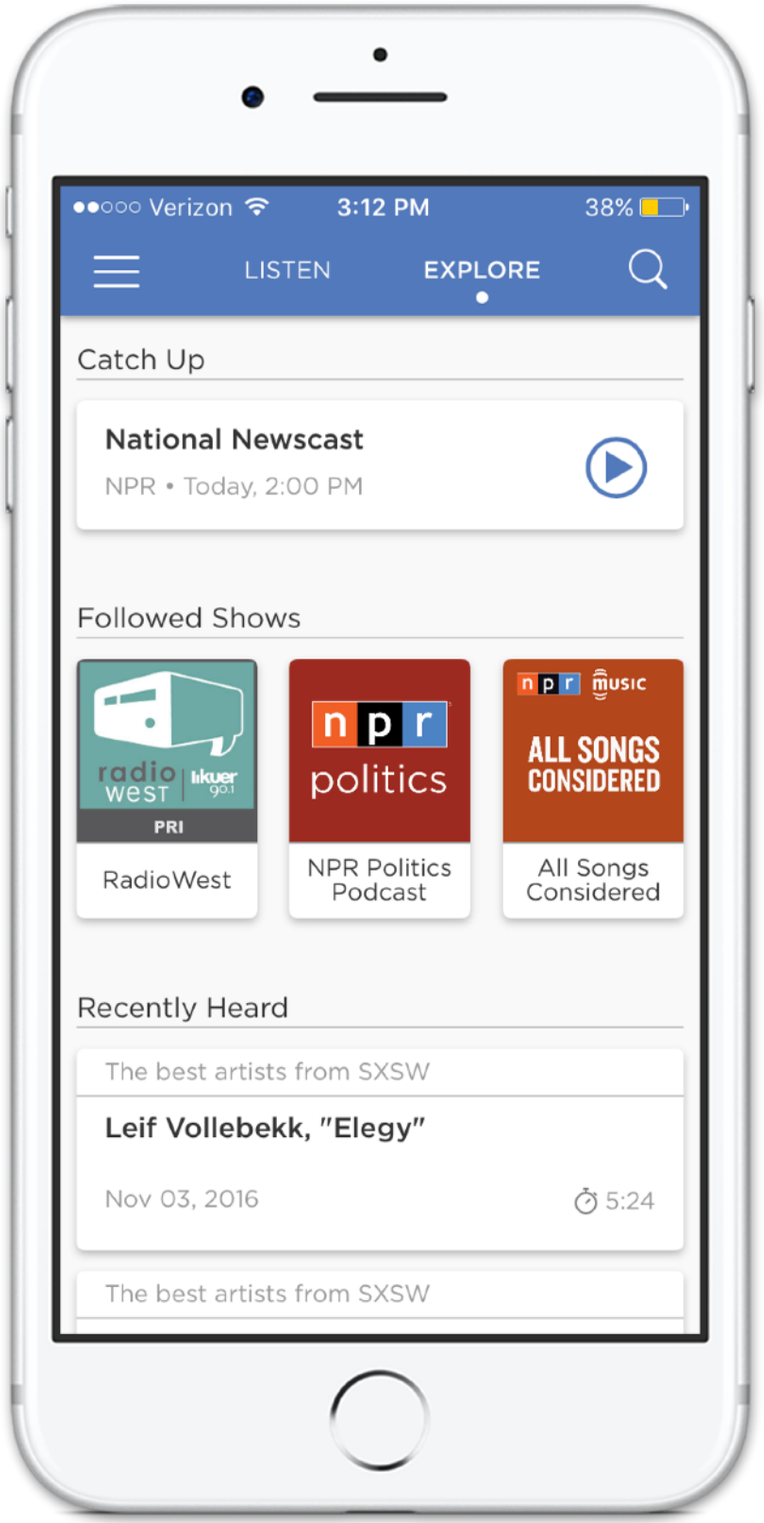


Image : NPR One

Core Data : Vocabulary

Data Model

Appears as a .**xcdatamodeled** file

Think of it as a spreadsheet

Entities and Attributes

Think of Entities as Classes or Objects, and attributes are the properties of those objects.

Example: For an app that stores a list of dog profiles, entities are a array of Dog objects, and Attributes are name, age, fur color, etc.

Core Data : Vocabulary

NSManagedObject

What an Entity appears as in our code

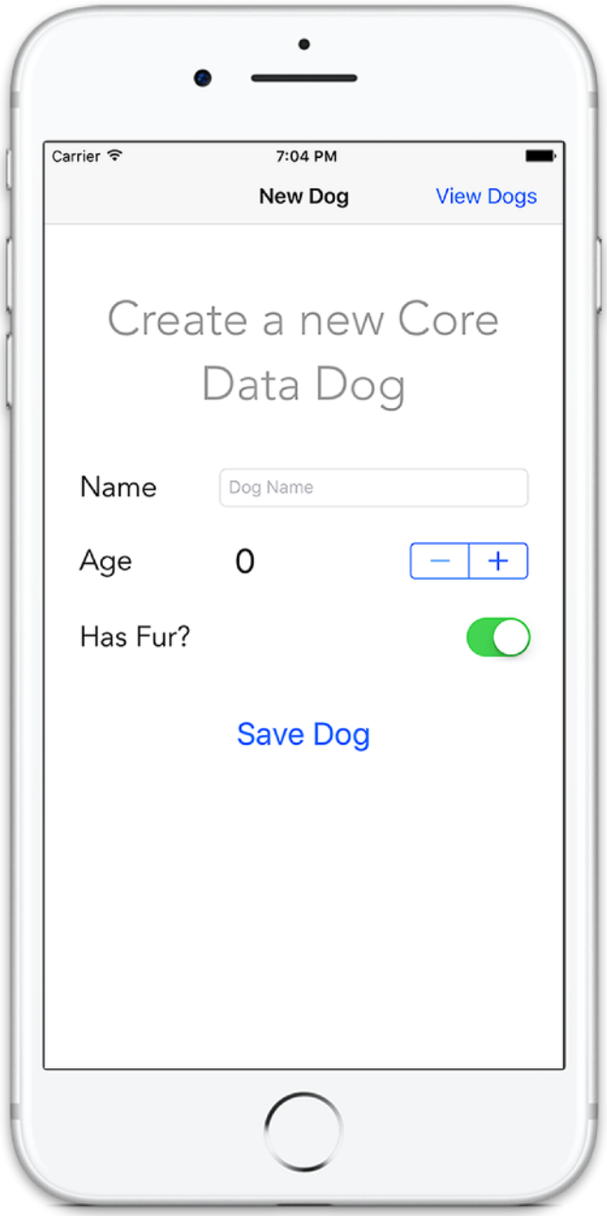
NSManagedObjectContext

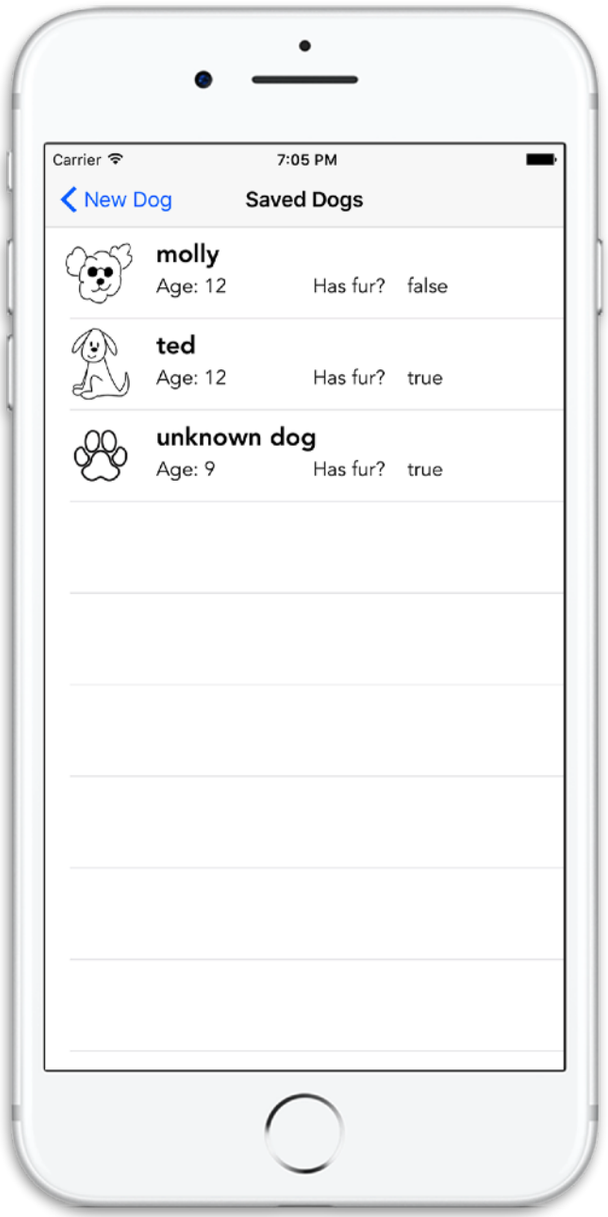
Allows you to access (store/query) data from the Container. Found in your App Delegate

AppDelegate.swift

Contains Core Data related methods and properties you'll need to interact with

Core Data : Today's Example





Allow user to add a name, age, and set whether or not dog has fur.

Using Core Data, save this user data to the user's device, so they can store a list of dogs

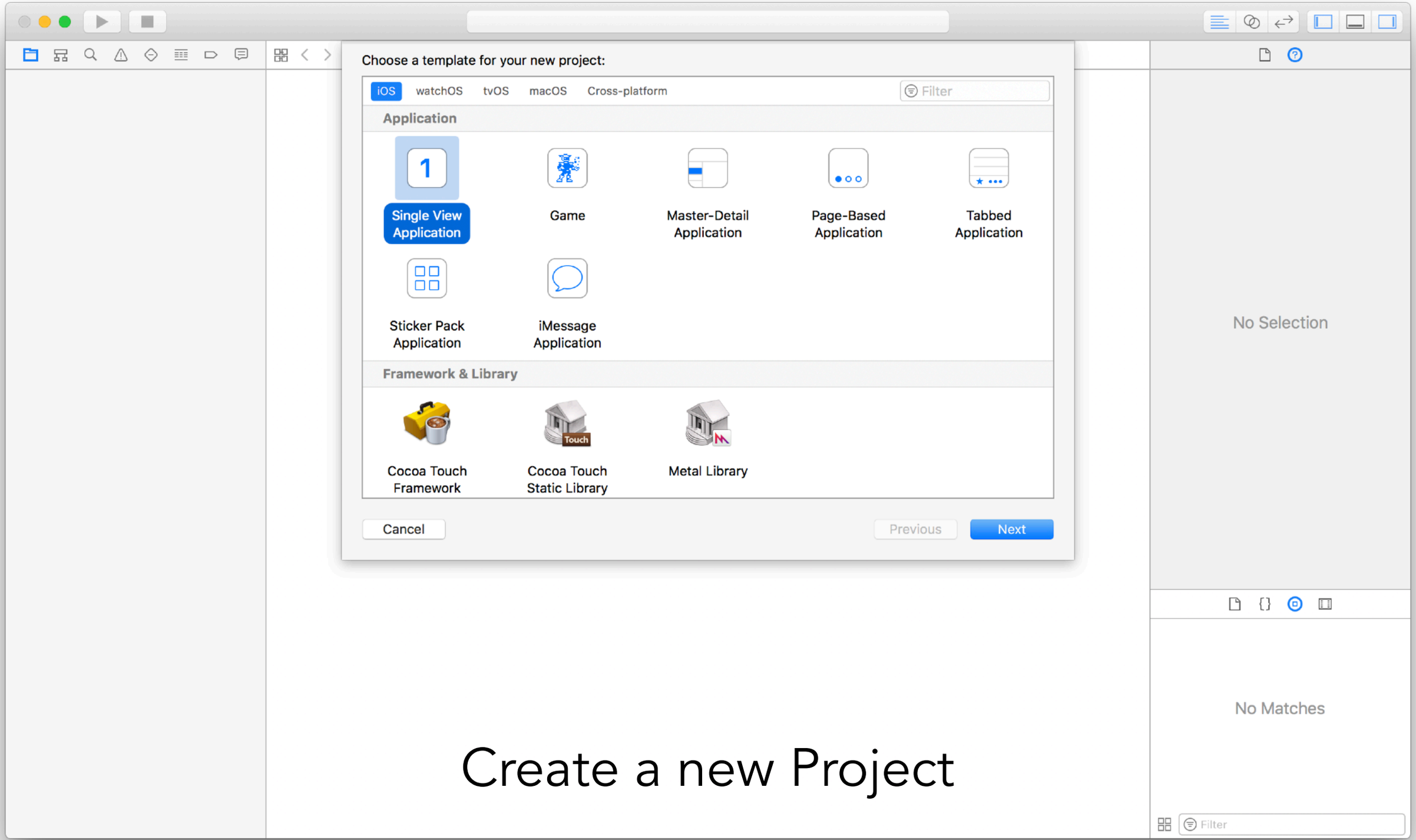
Code available at **github.com/paigeplan/Core-Data-Demo**

Core Data Checklist

1. **Enable Core Data:** Check **“Use Core Data”** when creating a new application
2. **Create an Entity:** for whatever you want to save the state for (i.e. Dog, Person, Profile)
3. **Store Data to Core Data:** Save user input data in an Entity

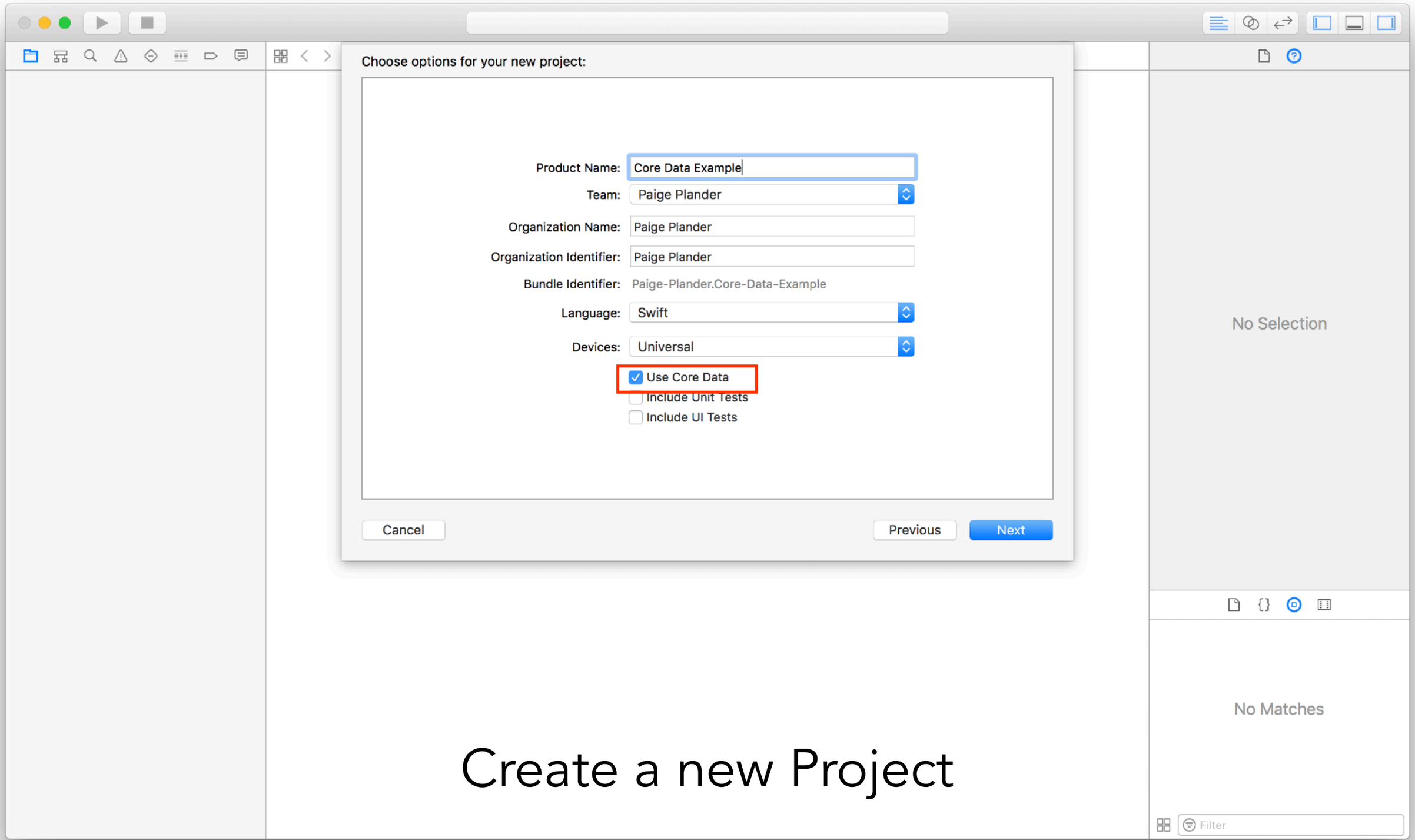
    This involves accessing `NSManagedObjectContext`, which you get through your App Delegate’s `NSPersistentContainer`
4. **Fetch Data from Core Data:** Access stored data using your context via the method `fetchRequest()`

Core Data : Enabling Core Data



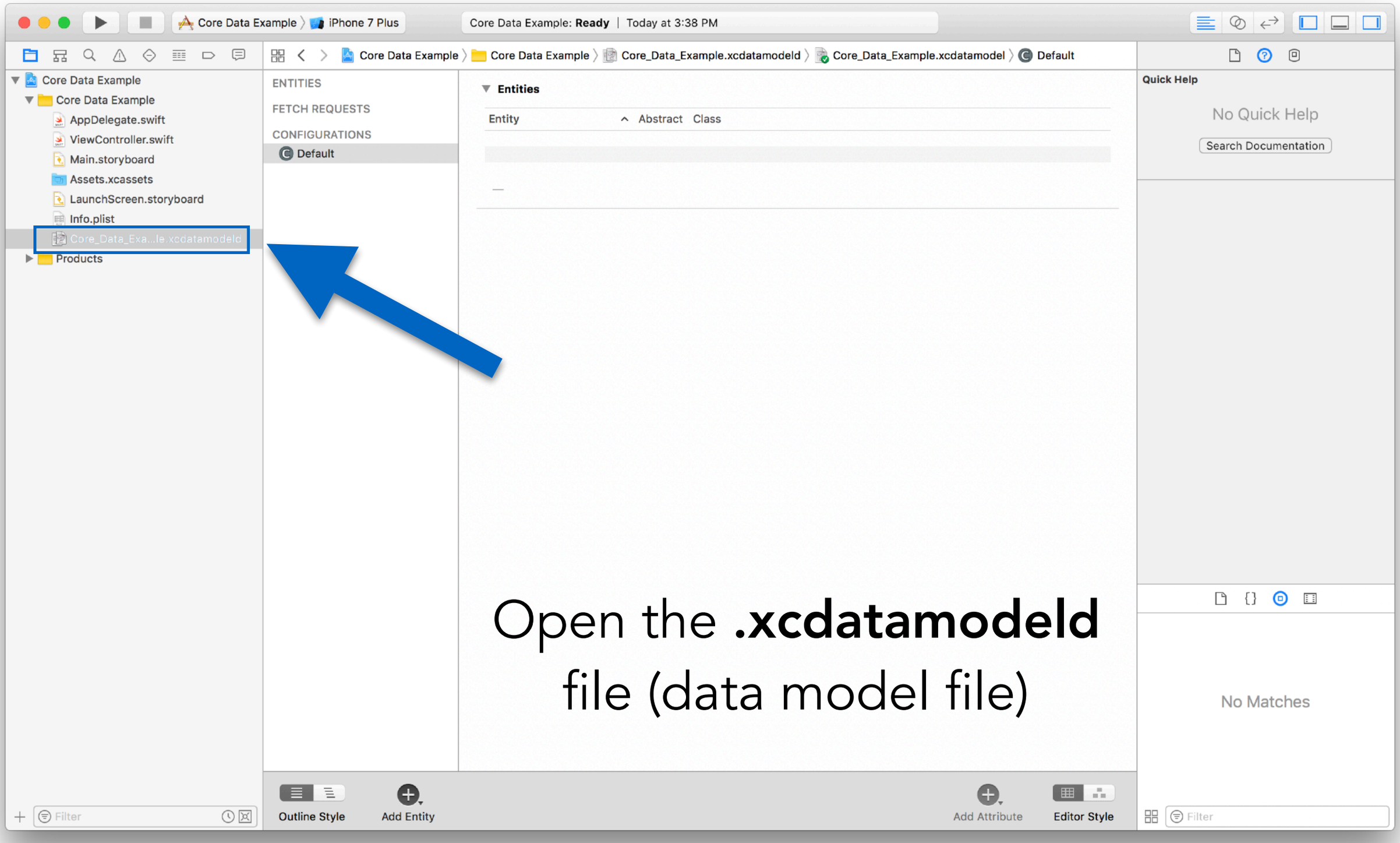
Create a new Project

Core Data : Enabling Core Data

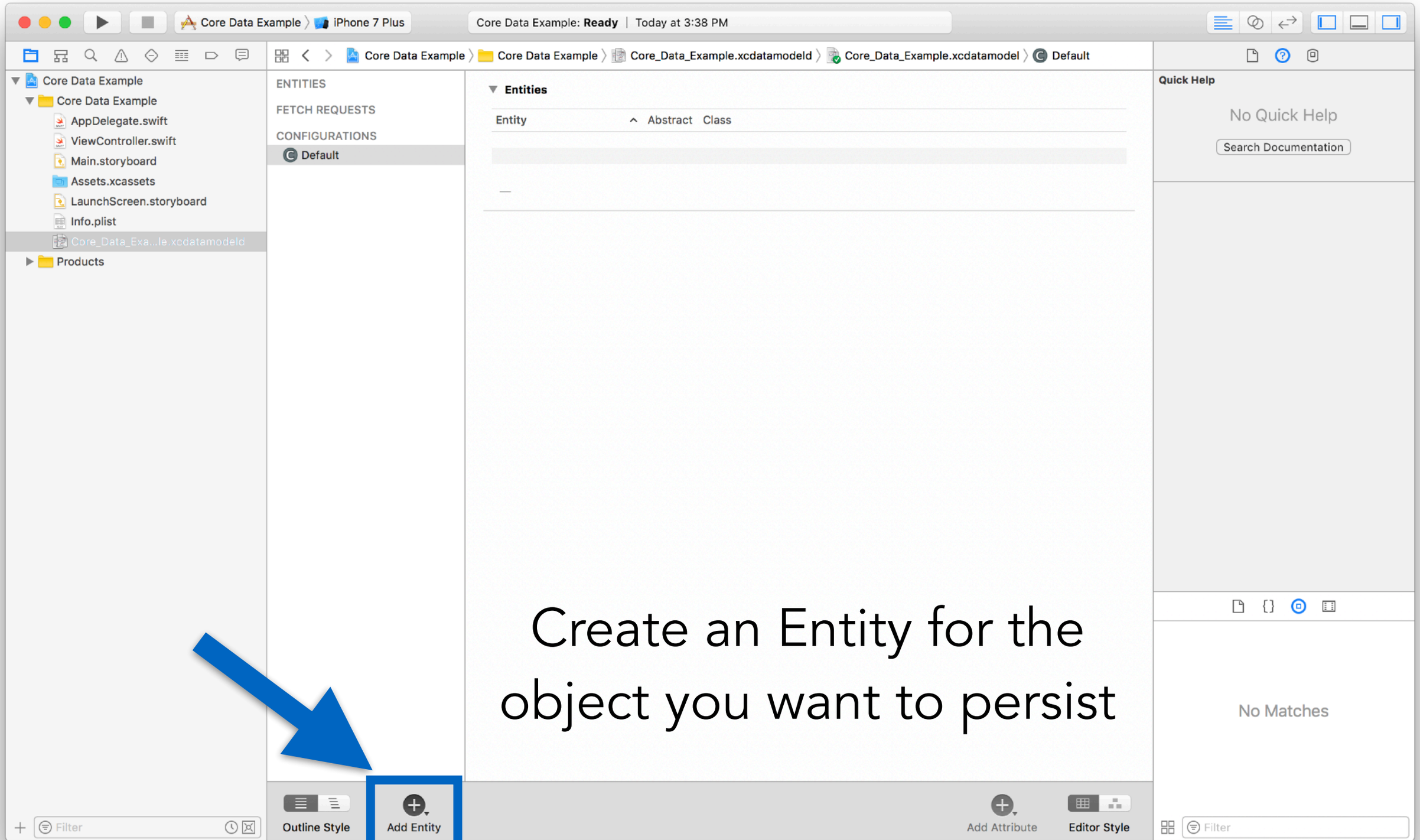


Create a new Project

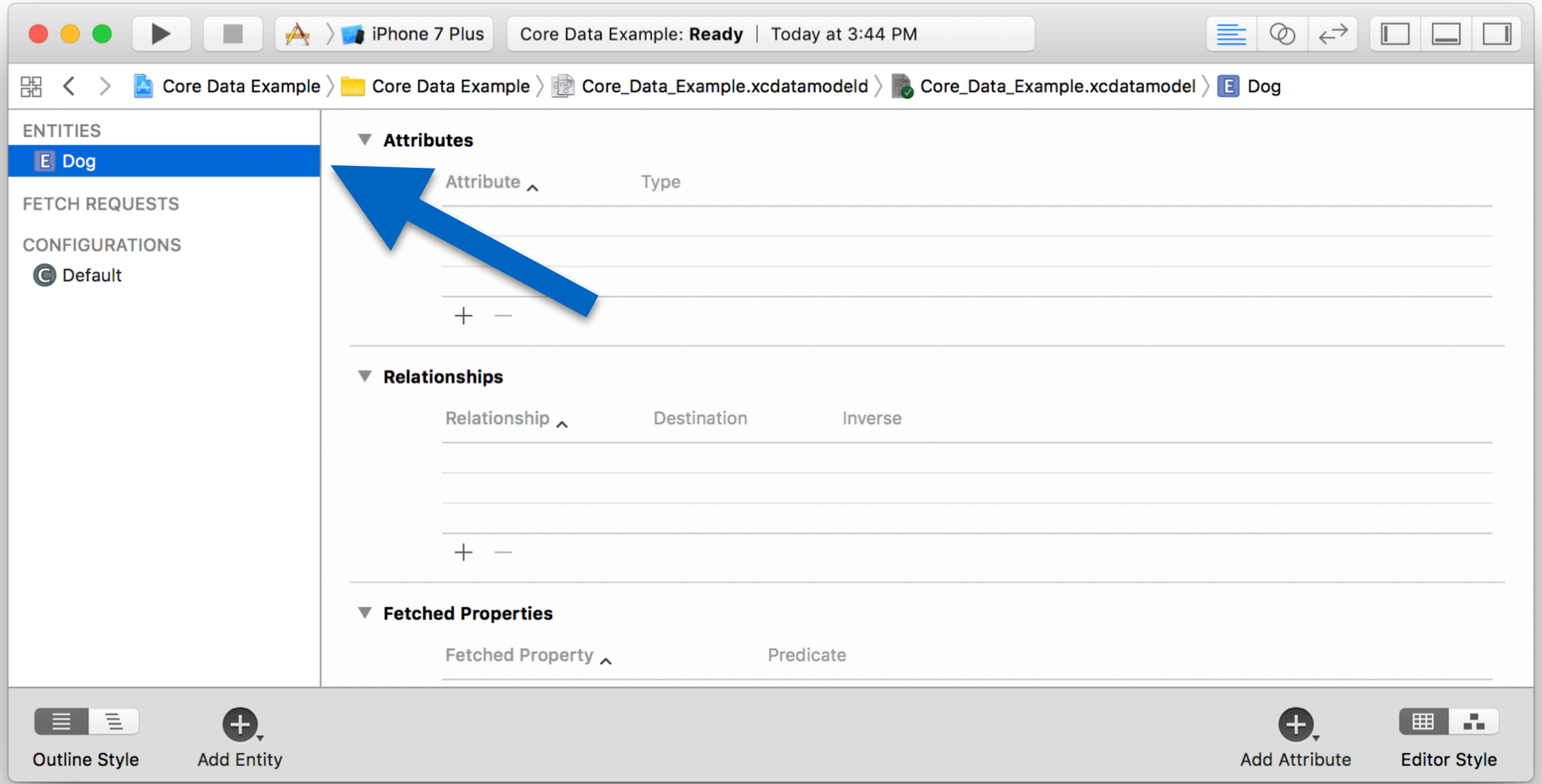
Core Data : Creating an Entity



Core Data : Creating an Entity

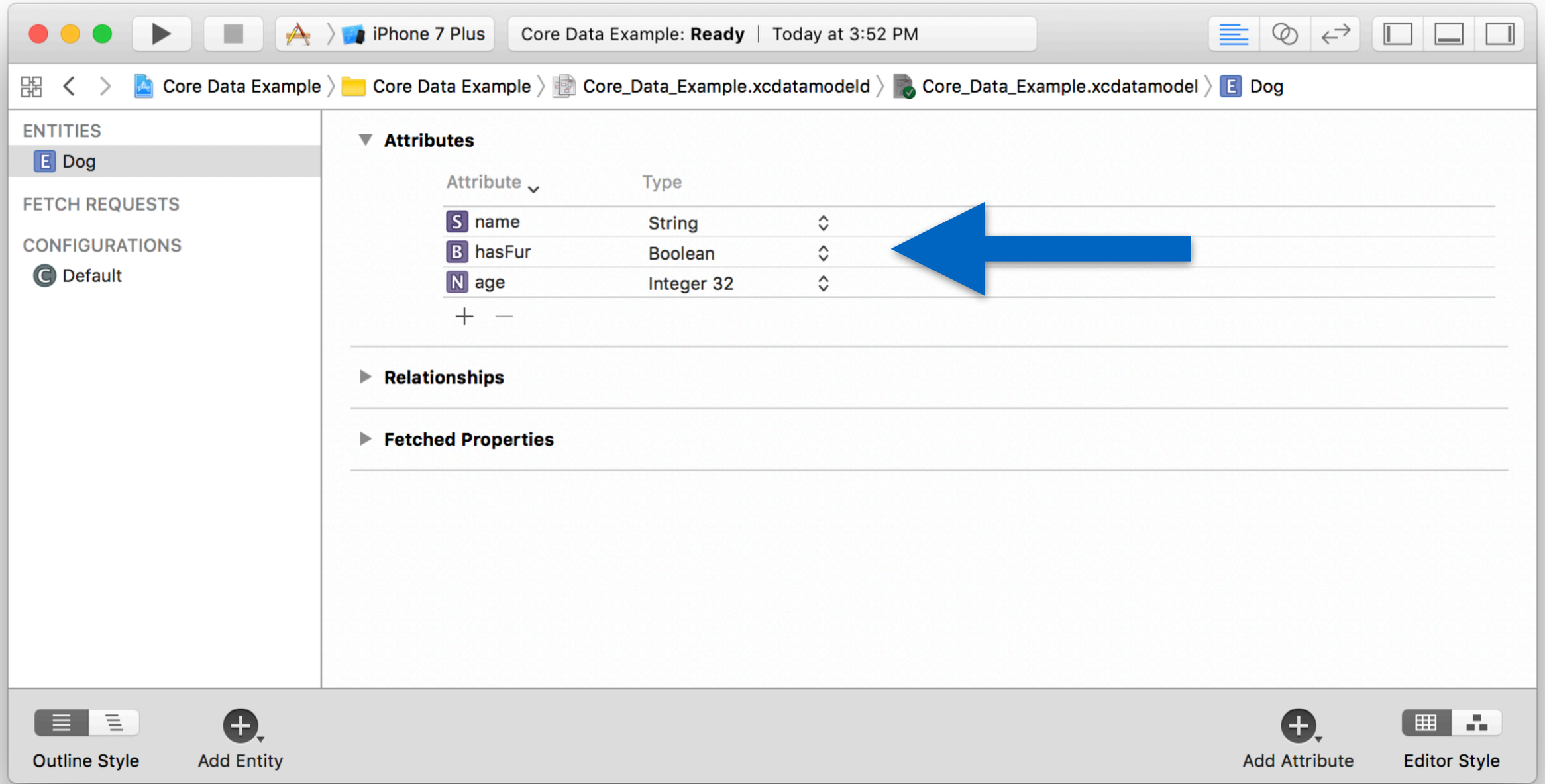


Core Data : Creating an Entity



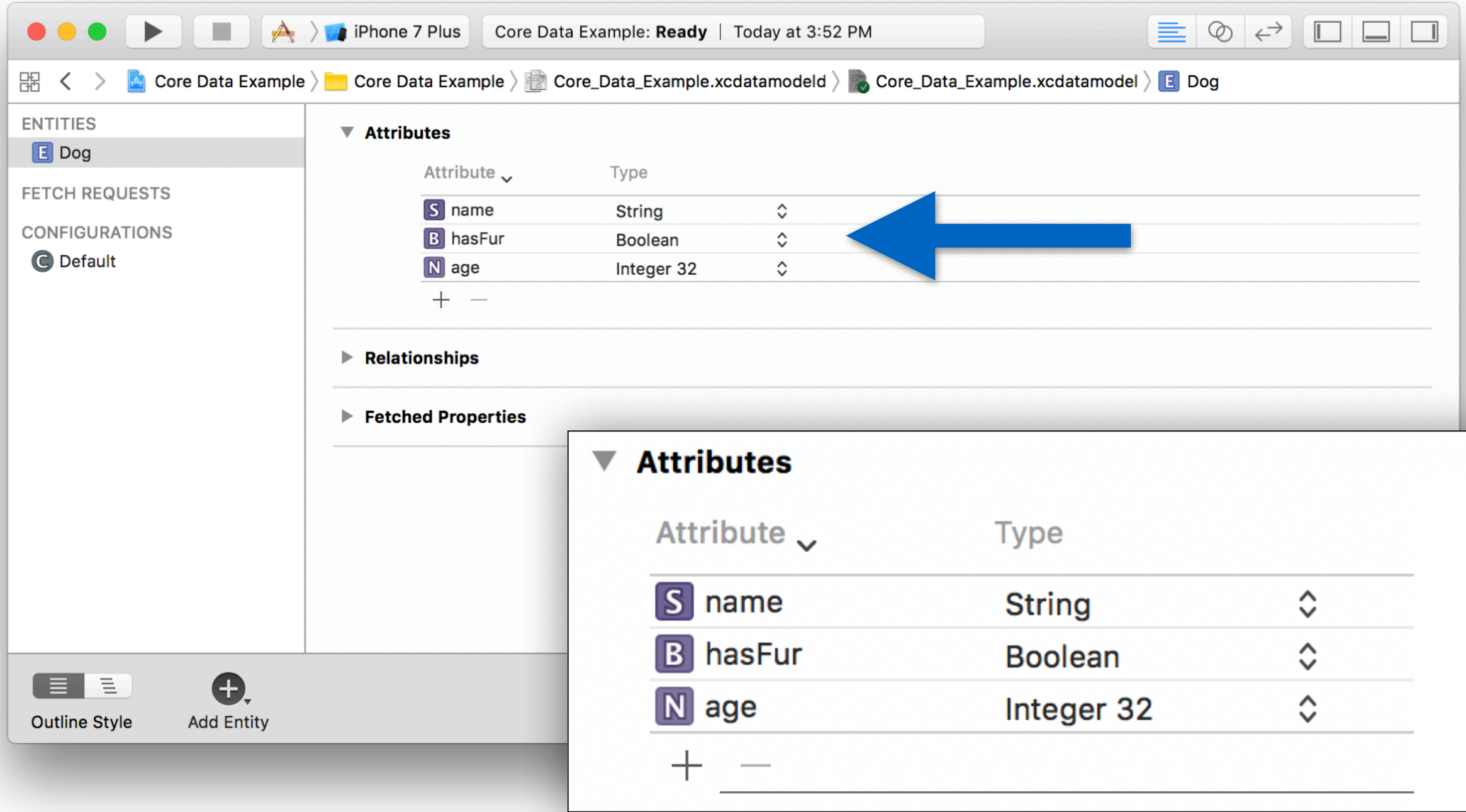
Name your entity (object)

Core Data : Creating an Entity



Add attributes (Model object instance variables)

Core Data : Creating an Entity



Add attributes (Model object instance variables)

Core Data : Storing Data

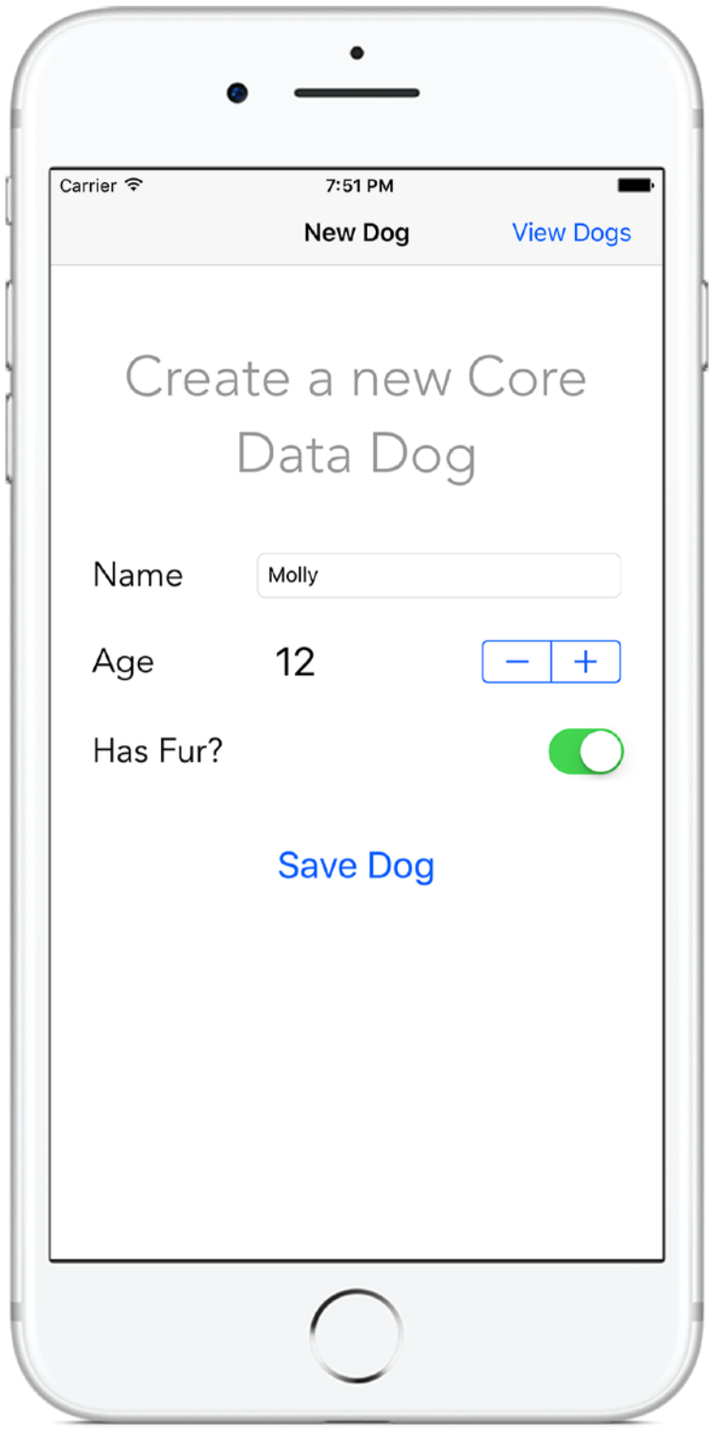


Figure out what user input that you want to save.

In this example, this involves getting the Name TextView's text, Age label text, and switch value

In the next slide, we'll store these values in Core Data

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

First, get a reference to your App Delegate

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

Get the context from the App Delegate. We need it to save our new Dog Object

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

Link **Dog** to **context**. **Dog** is a **NSManagedObjectContext**

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

Set the dog's **attributes**

Core Data : Storing Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
if let dogName = dogNameTextField.text {
    let dog = Dog(context: context)
    dog.name = dogNameTextField.text
    dog.hasFur = furSwitch.isOn
    dog.age = Int16(ageLabel.text)!
    appDelegate.saveContext()
}
```

Save **dog** to Core Data using `saveContext()`

Core Data : Retrieving Data

Now that we can store user data to Core Data, we need a way to retrieve this data so we can display / use it.

To do this we'll need to use our App Delegate and context again

Core Data : Retrieving Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
var dogs: [Dog] = []

func fetchDogsFromCoreData() {
    do {
        dogs = try context.fetch(Dog.fetchRequest())
    }
    catch {
        print(“Fetch failed :( ")
    }
}
```

Core Data : Retrieving Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
var dogs: [Dog] = []

func fetchDogsFromCoreData() {
    do {
        dogs = try context.fetch(Dog.fetchRequest())
    }
    catch {
        print(“Fetch failed :( ")
    }
}
```

Again, get App Delegate and context

Core Data : Retrieving Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
var dogs: [Dog] = []

func fetchDogsFromCoreData() {
    do {
        dogs = try context.fetch(Dog.fetchRequest())
    }
    catch {
        print(“Fetch failed :( ")
    }
}
```

Initialize an array to store your fetched Objects

Core Data : Retrieving Data

```
let appDel = UIApplication.shared.delegate
                                    as! AppDelegate
let context = appDel.persistentContainer.viewContext
var dogs: [Dog] = []

func fetchDogsFromCoreData() {
    do {
        dogs = try context.fetch(Dog.fetchRequest())
    }
    catch {
        print(“Fetch failed :( ")
    }
}
```

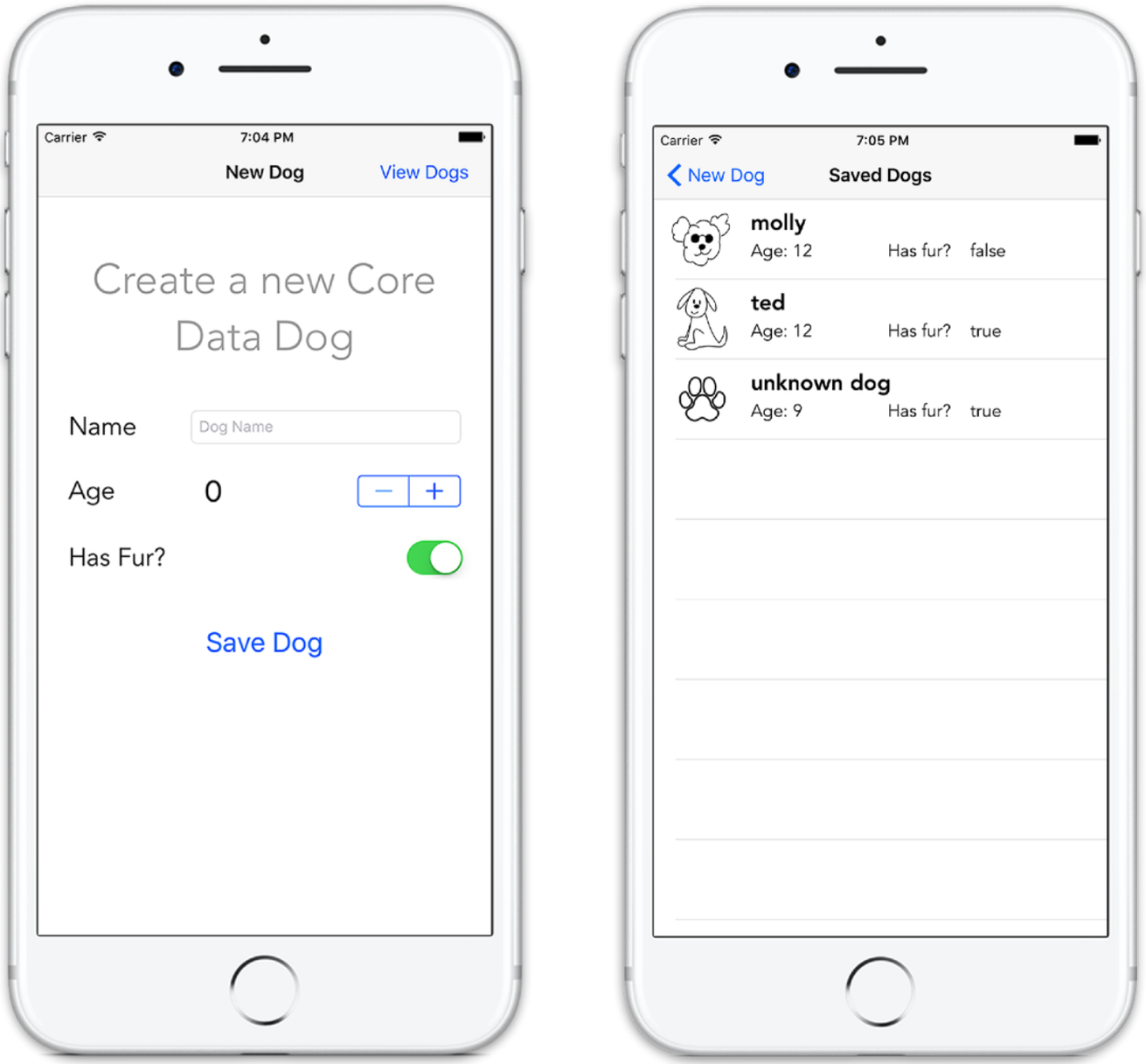
Populate this array with the Objects the user saved

Core Data : Retrieving Data

```
/// Uses the App Delegate's Context to get the dogs
saved to Core Data
func fetchDogsFromCoreData() {
  do {
     let request = NSFetchRequest<NSManagedObject>
                                        (entityName: "Dog")
     // only get 20 objects at a time
     myRequest.fetchBatchSize = 20
     // only give the first 100
     myRequest.fetchLimit = 100
     savedDogs = try context.fetch(myRequest) as! [Dog]
  } catch {
     print("Fetching Dogs from Core Data failed :( ")
  }
}
```

Can also set request limits / batch sizes if dealing with a lot of data

Core Data : Result!



Now the dogs the user has added will now be saved to disc.

We now don’t have to worry about data disappearing when the user force closes app or turns off phone

Proj 2 Pt 1 : Snapchat Clone

Due **next Tuesday** at 11:59pm

Lab 4 : Pokedex

Due **next Tuesday** at 11:59pm

Next Lecture: CocoaPods and Firebase