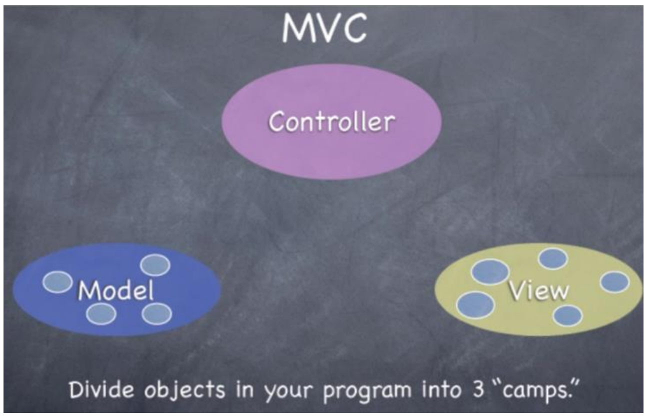
MVC IN IOS

- Model-View-Controller (MVC) is an exceptionally powerful software architectural pattern for creating iOS apps.
- MVC is the answer to the question: “How should I organize code in my iOS app?”
- The Model-View-Controller concept describes 3 components:



  - Model, a wrapper of data
  - View, a representation of a user interface (UI)
  - Controller, an intermediary between the Model and the View
- You use these 3 components together to structure your app. Every component has a distinct role:
  - The model represents the application's data and business logic. It is responsible for managing the application's data and ensuring that it remains consistent and up to date.
  - The view represents the user interface of the application. It is responsible for displaying the application's data to the user and capturing the user's input.
  - The controller acts as a mediator between the model and the view. It is responsible for interpreting user input, updating the model accordingly, and updating the view to reflect any changes in the model.

- In iOS development, the UIKit framework provides a range of classes that can be used to implement the MVC pattern.
- For example, UIView represents the view, UIViewController represents the controller, and various data objects can represent the model.

- It's important to note that while the MVC pattern is commonly used in iOS development, it's not the only design pattern that can be used. Other patterns, such as MVVM (Model-View-ViewModel), can also be used to achieve similar separation of concerns. Ultimately, the choice of design pattern will depend on the specific needs of the application being developed.

CAN THE VIEW CAN COMMUNICATE WITH THE CONTROLLER?

In an“indirect”way yes for iOS

- STEP 1: Controller drops a target on itself called an“outlet”
- STEP2: Give the View an “action” that can be triggered to communicate to the “outlet” of the Controller
- STEP 3: View when altered (say by user doing something in it—i.e. typing text) through the “action” notifies the “outlet” of change

Jailbreaking

- The process of removing limitations on iOS, Apple's operating system, on devices running it through the use of software and hardware exploits.
- Target: iPhone, iPod touch, iPad and second-generation Apple TV.
- **Why is “Jailbreak” ?**
- allowing the installation of software that is unavailable through the official Apple App Store
- A jailbroken running iOS can still use the App Store, iTunes, and other normal functions.
- Potentials of Security, Privacy and Stabilit

What is iOS ?

iOS (formerly iPhone OS) is Apple’s mobile operating system.

Apple does not license I for installation on non-Apple hardware

(distributed exclusively for Apple hardware)

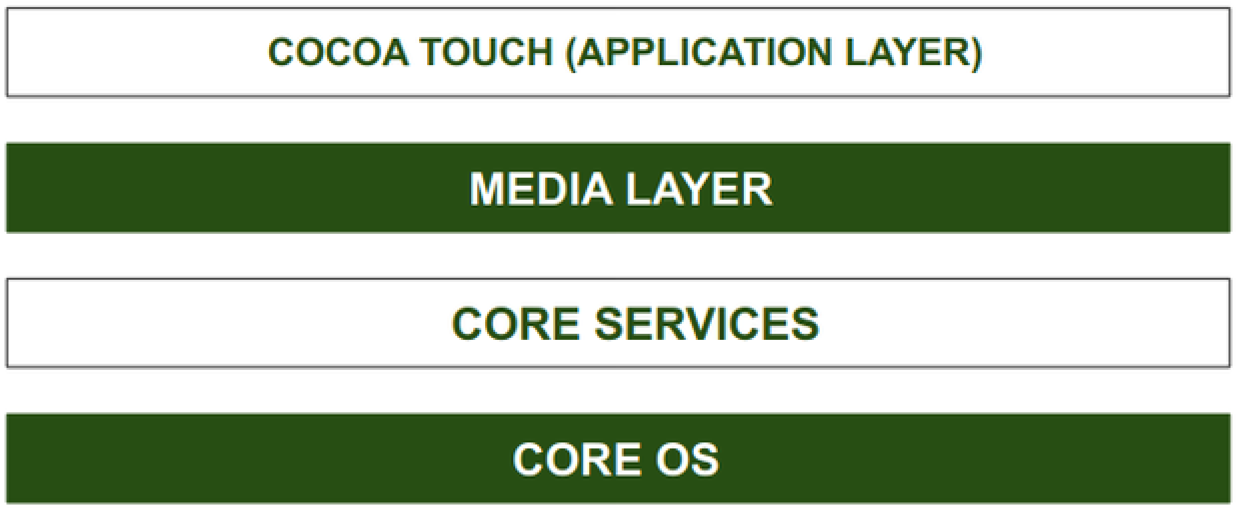
OS is derived from Mac OS X.

- Originally developed for the iPhone, but it’s extended now to support other Apple devices such as the iPod Touch, iPad, and Apple TV

Architecture of IOS Operating System

- IOS is a Mobile Operating System that was developed by Apple Inc. for iPhones, iPads, and other Apple mobile devices.
- iOS is the second most popular and most used Mobile Operating System after Android.
- The structure of the iOS operating System is Layered based.
- Its communication doesn't occur directly.
- The layer's between the Application Layer and the Hardware layer will help for Communication.
- The lower level gives basic services on which all applications rely and the higher-level layers provide graphics and interface-related services.
- Most of the system interfaces come with a special package called a framework.

- A framework is a directory that holds dynamic shared libraries.
- Each layer has a set of frameworks that are helpful for developers.



Architecture of IOS

- **CORE OS Layer:**

    - All the IOS technologies are built under the lowest level layer i.e. Core OS layer INCLUDE technologies:
        1. Core Bluetooth Framework
        2. External Accessories Framework
        3. Accelerate Framework

4. Security Services Framework
5. Local Authorization Framework etc.

- It supports 64 bit which enables the application to run faster.

CORE SERVICES Layer:

- Some important frameworks are present in the CORE SERVICES Layer which helps the iOS operating system to cure itself and provide better functionality.
-
- It is the 2nd lowest layer in the Architecture as shown above.
- Below are some important frameworks present in this layer:

1. **Address Book Framework-**
   The Address Book Framework provides access to the contact details of the user.
2. **Cloud Kit Framework-**
   This framework provides a medium for moving data between your app and iCloud.
3. **Core Data Framework-**
   This is the technology that is used for managing the data model of a Model View Controller app.
4. **Core Foundation Framework-**
   This framework provides data management and service features for iOS applications.
5. **Core Location Framework-**
   This framework helps to provide the location and heading information to the application.
6. **Core Motion Framework-**
   All the motion-based data on the device is accessed with the help of the Core Motion Framework.
7. **Foundation Framework-**
   Objective C covering too many of the features found in the Core Foundation framework.
8. **HealthKit Framework-**
   This framework handles the health-related information of the user.

9. **HomeKit Framework-**
   This framework is used for talking with and controlling connected devices with the user's home.
10. **Social Framework-**
    It is simply an interface that will access users' social media accounts.
11. **StoreKit Framework-**
    This framework supports for buying of contents and services from inside iOS apps.

MEDIA Layer:

- With the help of the media layer,
  we will enable all graphics video, and audio technology of the system.
- This is the second layer in the architecture.
- The different frameworks of MEDIA layers are:

1. **ULKit Graphics-**
   This framework provides support for designing images and animating the view content.
2. **Core Graphics Framework-**
   This framework support 2D vector and image-based rendering and it is a native drawing engine for iOS.
3. **Core Animation-**
   This framework helps in optimizing the animation experience of the apps in iOS.
4. **Media Player Framework-**
   This framework provides support for playing the playlist and enables the user to use their iTunes library.
5. **AV Kit-**
   This framework provides various easy-to-use interfaces for video presentation, recording, and playback of audio and video.
6. **Open AL-**
   This framework is an Industry Standard Technology for providing Audio.

7. **Core Images-**
   This framework provides advanced support for motionless images.
8. **GL Kit-**
   This framework manages advanced 2D and 3D rendering by hardware-accelerated interfaces.

- **COCOA TOUCH:**
  - COCOA Touch is also known as the application layer which acts as an interface for the user to work with the iOS Operating system.
  - It supports touch and motion events and many more features.
  - The COCOA TOUCH layer provides the following frameworks :

1. **EvenKit Framework-**
   This framework shows a standard system interface using view controllers for viewing and changing events.
2. **GameKit Framework-**
   This framework provides support for users to share their game-related data online using a Game Center.
3. **MapKit Framework-**
   This framework gives a scrollable map that one can include in your user interface of the app.
4. **PushKit Framework-**
   This framework provides registration support.

Features of iOS operating System:

Let us discuss some features of the iOS operating system-

1. Highly Securer than other operating systems.
2. iOS provides multitasking features like while working in one application we can switch to another application easily.
3. iOS's user interface includes multiple gestures like swipe, tap, pinch, Reverse pinch.
4. iBooks, iStore, iTunes, Game Center, and Email are user-friendly.
5. It provides Safari as a default Web Browser.
6. It has a powerful API and a Camera.
7. It has deep hardware and software integration

Applications of IOS Operating System:

Here are some applications of the iOS operating system-

1. iOS Operating System is the Commercial Operating system of Apple Inc. and is popular for its security.
2. iOS operating system comes with pre-installed apps which were developed by Apple like Mail, Map, TV, Music, Wallet, Health, and Many More.
3. Swift Programming language is used for Developing Apps that would run on IOS Operating System.
4. In iOS Operating System we can perform Multitask like Chatting along with Surfing on the Internet.

Advantages of IOS Operating System:

The iOS operating system has some advantages over other operating systems available in the market especially the Android operating system. Here are some of them-

1. More secure than other operating systems.
2. Excellent UI and fluid responsive
3. Suits best for Business and Professionals
4. Generate Less Heat as compared to Android.

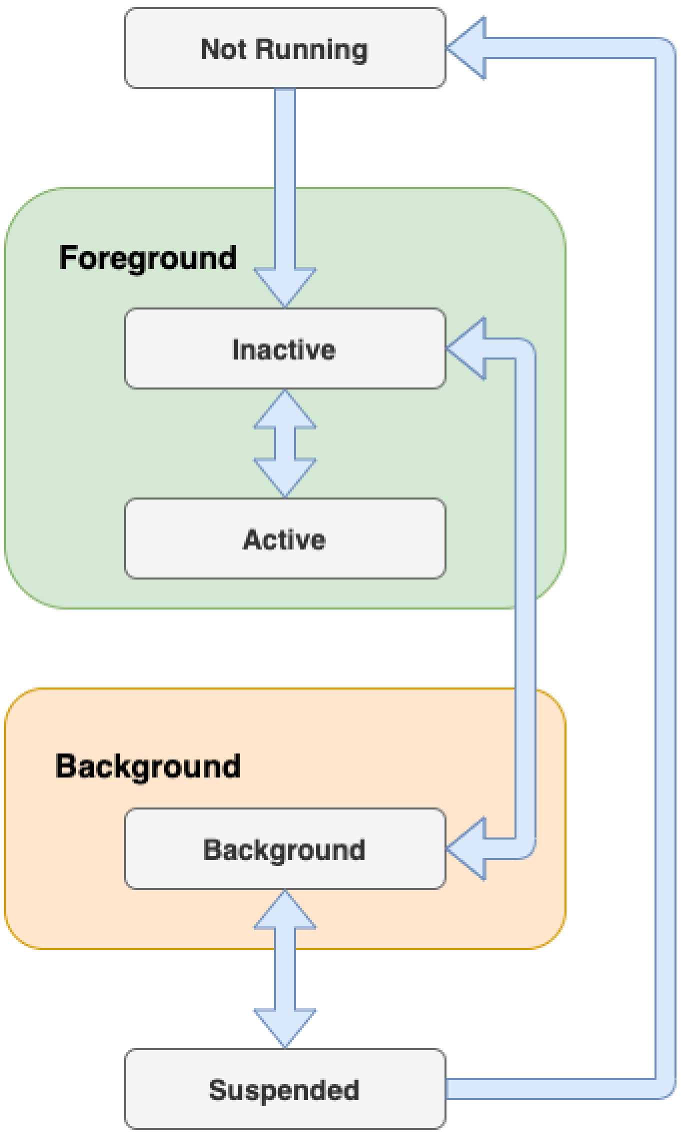
Disadvantages of IOS Operating System:

Let us have a look at some disadvantages of the iOS operating system-

1. More Costly.
2. Less User Friendly as Compared to Android Operating System.
3. Not Flexible as it supports only IOS devices.
4. Battery Performance is poor.

IOS app lifecycle

- An iOS application runs into several states,
- which are called the state of the Application life cycle.
- Every iOS developer must be aware of the app life cycle,
- which helps to understand the application's behavior.
- Every iOS application passes through the following states as it runs.



1. **Not Running:** the app is considered to be in a Not Running state when it is not yet launched or terminated by the system or user.
2. **Inactive:** the app is in an inactive state when it is in the foreground but receiving events. In other words, we can say that it acts like a bridge state in which the app remains briefly when it transitions to a different state.
3. **Active:** it is a normal mode for the app when it is in the foreground state and receiving all the user events.

4. **Background:** the app transitions into the background state when the user taps on the home screen while using the application, or it requires some extra execution time. When the app is about to be suspended, then also transitions into this state for a small amount of time. In this state, the app remains in the background and executes the code.
5. **Suspended:** in this state, the app remains in the background and doesn't execute the code. The app is automatically moved to this state. In this state, the app remains in memory. However, the foreground apps are always given priority over suspended apps and can be purged any time without notice.

- We must notice that when we build and run an iOS application in XCode,
- the main entry point of the application is UIApplicationDelegate,
- which is a protocol that the application must implement to get notified of several user events like app launch, the app goes into the background, the app goes to the foreground, push notifications, etc.
- The UIApplicationDelegate contains certain app lifecycle methods that are notified when the app starts running.
- The UIApplicationDelegate methods are given below.

1. **application: didFinishLaunchingWithOptions:-> Bool:** when the application is launched initially, this method is called. We can do any initial setup for the application in this method like firebase configuration, user navigation, etc. The storyboard is loaded at this point, but we can maintain the state restoration.
2. **applicationWillEnterForeground:** this method is called after didFinishLaunchingWithOptions. It is also called when the application comes from background to foreground.
3. **applicationDidBecomeActive:** this method is called after applicationWillEnterForeground. If we need to perform any particular task when the app comes into the foreground, like font update, etc., then we can place the code in this method.
4. **applicationWillResignActive:** this method is notified when the application is about to become inactive. For example, the user receives a phone call; the user presses the home button, etc.).
5. **applicationDidEnterBackground:** this method is notified when the application goes into a background state after become inactive.

6. **applicationWillTerminate:** this method is called when the application is about to be finally terminated from memory. If we need to perform any final cleanups, then we can place the code in this method.

- We can place the code which is to be run on app launch in **didFinishLaunchingWithOptions** method.
- For example, in an application in which user login is required to use the app, we can check if the user is already logged in to the application via checking UserDefaults.
- If the user is logged in, then we can navigate the user to the home screen; otherwise, we can navigate it to the login screen.

Core Location

- Core Location is a framework that allows applications to retrieve the location and heading of the device they are running on.
- To do this, Core Location can use a combination of a compass for heading, and either GPS, cellular radio, or WiFi technologies for location.
- Cellular radio and WiFi-based location is less accurate than GPS.
- Applications cannot specify which method will be used, but they can specify a desired
- level of accuracy. Depending on the desired level of accuracy, Core Location tries to use the GPS hardware, cellular radio, or WiFi in that order.
- This framework is not included in any of the standard iOS application templates.
- To use this framework in your code, you will need to add it manually to your project.
- You can do this from the Project Settings page in Xcode.
- Select the project node in the project navigator to display the settings page.
- On the settings page, switch to the Build Phases tab and click the + button under the Link Binary With Libraries category.
- Select CoreLocation.framework
- from the list of available frameworks.
- Core Location defines a manager class called CLLocationManager that you can use to interact with the framework.
- It allows you to specify the desired frequency and accuracy of location information.
- To receive location updates in an application, you need to create an instance of the CLLocationManager class and provide a delegate object to receive location updates and errors.

- This delegate object must implement the CLLocationManagerDelegate protocol.
- The delegate object is often a view controller class but could also be any other class in your application. Using location hardware can have a significant drain on the device's batteries, and hence applications need to turn on and turn off receiving location updates.

PERMISSIONS

- From iOS 8 onward, Apple requires that applications ask the user for permission before attempting to access location information.
- There are two types of permissions available:

  ➤**Always Authorization**: For apps that need to access location information while in both the foreground and background modes

  ➤**When In Use Authorization**: For apps that need only access location information in the foreground mode

- You need to ask for either type of permission, but not both. The process of asking for permission has two parts.
- The first part involves adding a key to the Info.plist file that contains some text that will be presented to the user while asking for permission.
- This text should describe the reason for application requiring access to location data.
- Depending on the type of permission you wish to ask for, you need to add either of the following keys to the Info.plist file.
  - NSLocationAlwaysUsageDescription
  - NSLocationWhenInUseUsageDescription

ACCURACY

- An application can set up the desiredAccuracy property of the CLLocationManager instance to specify a desired accuracy.
- Core Location will try its best to achieve the desired accuracy.
- The more accurate a reading required, the more battery power is needed.
- Applications should, in general, try to use the least accuracy possible to satisfy their requirements.
- The property can have the following values, listed in decreasing order of accuracy:
- kCLLocationAccuracyBestForNavigation
- kCLLocationAccuracyBest
- kCLLocationAccuracyNearestTenMeters
- kCLLocationAccuracyHundredMeters
- kCLLocationAccuracyKilometer
- kCLLocationAccuracyThreeKilometers

- An application can also set up the distanceFilter property of the CLLocationManagerinstance to specify the minimum distance in meters a device must move before an update is provided to the application.

- The default value of this property is kCLDistanceFilterNone,
- which specifies the application wants to know of all movements.

GEOCODING AND REVERSE GEOCODING

- Geocoding involves converting between a latitude/longitude coordinate pair and an address.
- Core Location provides the CLGeocoder class that provides methods to perform both forward and reverse geocoding.
- Forward-geocoding involves converting from an address to a latitude/longitude value.
- Reverse-geocoding involves converting a latitude/longitude value into an address.
- The result of a geocoding request is represented by a CLPlacemark object.
- A forward-geocoding request returns an array of CLPlacemarkobjects because multiple results may be returned.
- To perform a forward-geocoding request from an address string, you can call the geocodeAddressString(addressString: String, completionHandler: CLGeocodeCompletionHandler)method on a geocoder instance.
- This message requires you to specify a String object that contains an address string and a block handler that is called when the geocoding operation is complete.
- You can send the geocoder a reverse-geocoding request by calling the reverseGeocodeLocation(location: CLLocation, completionHandler: CLGeocodeCompletionHandler) method.
- A CLPlacemark object contains several properties that encapsulate information on an address associated with a specific coordinate.
- Some of the properties are:

   location: A CLLocation object that provides the coordinate pair associated with the placemark

 ISOcountryCode: An NSString object that contains the abbreviated country code

 country: An NSString object that contains the name of country

 postalCode: An NSString object that contains the postal code

 administrativeArea: An NSString object that contains the state/province

 locality: An NSString object that contains the city

 thoroughfare: An NSString object that contains the street address

 subThoroughfare: An NSString object that contains additional street address information

SQLLite

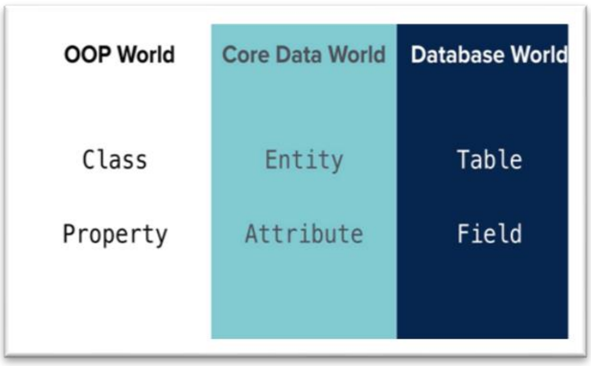
- The database that can be used by apps in iOS (and also used by iOS) is called SQLite, and it's a relational database.
- It is contained in a C-library that is embedded to the app that is about to use it.
- Note that it does not consist of a separate service or daemon running on the background and attached to the app.
- On the contrary, the app runs it as an integral part of it.
- Now a days, SQLite lives its third version, so it's also commonly referred as SQLite 3.
- Features:
    - It's lightweight.
    - It contains an embedded SQL engine, so almost all of your SQL knowledge can be applied.
    - It works as part of the app itself, and it doesn't require extra active services.
    - It's very reliable.
    - It's fast.
    - It's fully supported by Apple, as it's used in both iOS and MacOS.
    - It has continuous support by developers in the whole world and new features are always added to it

The Core SQLite functions

- Lets first start with a list of the most used SQLite functions and describe their purpose:

- **sqlite3_open()**: This function creates and opens an empty database with the specified filename argument. If the database already exists it will only open thedatabase. Upon return the second argument will contain a handle to the database instance.
- **sqlite3_close():** This function should be used to close a previously openedSQLite database connection. It will free all system resources associated with the database connection.
- **sqlite3_prepare_v2():** To execute an SQL statement it first needs to be compiled into byte-code and that is exactly what this function is doing. It basically transforms an SQL statement written in a string to an executable piece of code.
- **sqlite3_step():** Calling this function will execute a previously prepared SQL statement.
- **sqlite3_finalize():** This function deletes a previously prepared SQL statement from memory.
- **sqlite3_exec():** Combines the functionality of sqlite3_prepare_v2(), sqlite3_step() and sqlite3_finalize() into a single function call.
- **sqlite3_column_<type>():** This routine returns information about a single column of the current result row of a query. Typical values for <type> are text and int.It is important to note that the column indexes are zero based

CORE DATA



OOP World	Core Data World	Database World
Class	Entity	Table
Property	Attribute	Field

- framework that managing the lifecycle of objects in your application
- provides solutions to object serialization
- Prior to Core Data, programmers relied on SQLite
- It provides a convenient mechanism to create, update, and delete entities in the database without having to write a single line of SQL.
- based on the Model-View-Controller pattern
- Introduces few new concepts and terminology

Managed Object IN COREDATA

- a representation of the object that you want to save to the data store.
- similar to a record in a relational database table and typically contains Fields thatcorrespond to properties
- lifecycle of managed objects is managed by Core Data
- Managed objects are subclasses of NSManagedObjec

Managed Object Context in CORE Data

- is like a buffer between your application and the data store.
- contains all your managed objects before they are written to the data store and manages their lifecycles
- Inside this context you can add, delete, or modify managed objects

• To do these you will call methods on the managed object context Is an instance of the NSManagedObjectContext class.

• An application can have multiple managed object context

Persistent Store Coordinator in COREDATA

• is an instance of NSPersistentStoreCoordinator and represents the connection to the data store.

• contains low-level information, such as the name,location, and type of the data store to be used

• application will have one instance of the persistent store coordinator for each database that it needs to interact with.

• used by the managed object context Multiple managed object contexts can share the same instance of the persistent store coordinator

• Core Data handles the synchronization of data across all these contexts

Entity Description

• an instance of NSEntityDescription and describes a table within the database

• database tables are called entities

• to create entity descriptions you can use the graphical Core Data editor included within XCode

• similar to the schema for a database table

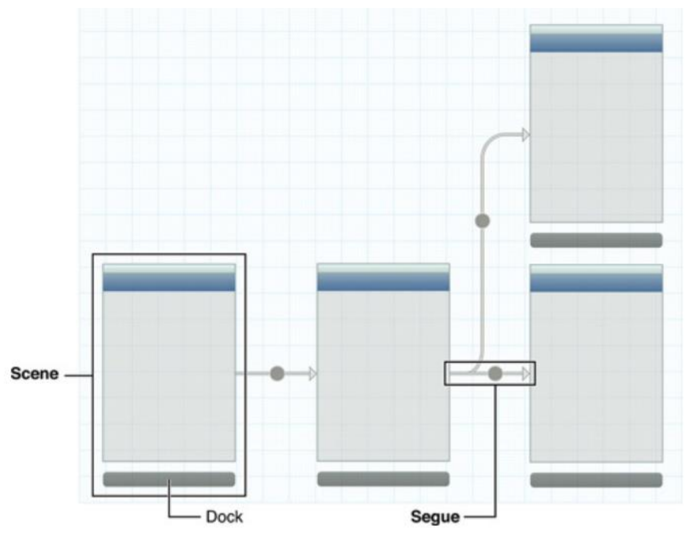
• It does not contain the actual data; it is used internally by Core Data to create tables in the underlying database

Managed Object Model

- an instance of NSManagedObjectModel and is a collection of entity descriptions
- a file that ends with the extension .xcdatamodeld
- This is used by XCode to build a graphical editor for the managed object model
- When project is compiled, this file is compiled into a .mom file, which is the managed object model in binary format
- keep in mind that this file will be compiled to produce the managed object model

What is Storyboard?

- A storyboard is a visual representation of the user interface of an iOS application, showing screens of content and the connections between those screens.
- A storyboard is composed of a sequence of scenes, each of which represents a view controller and its views; scenes are connected by segue objects, which represent a transition between two view controllers.
- Storyboard help us to create all the screens of the application along with their flow of screens under one interface MainStoryboard.storyboard.



- **Advantages of Storyboard**
    - You have better conceptual overview of all the screens and connections between them.
    - It describes the transition between the various screens.
    - Make working with table and views a lot easier with the new prototype and static cells features.
    - Can also use .xibs if needed.

- **Dis-advantages of Storyboard**
    - Need a big monitor.
    - Only available in iOS 5 onward
    - Merges can be very difficult