Harisha K N IOS Weekly evaluation 19-Mar-16

1. list various features of iOS Architecture, and explain any 3

Apple describes the set of frameworks and technologies that are currently implemented within the iOS operating system as a series of layers. Each of these layers is made up of a variety of different frameworks that can be used and incorporated into our applications.

Following are the main features of iOS Architecture. It is given in the below figure.

Cocoa Touch
Media
Core Service
Core OS

1.Cocoa Touch Layer

Cocoa Touch Layer contains the key frame work for building ios apps. It also provides basic app infrastructure such as multitasking, push notification, and touch based input and many high level services for the app.

The following are the key technologies available in the cocoa touch layer.

- **App Extension:** An app extension lets you extend custom functionality and content beyond your app and make it available to users while they're using other apps or the system. You create an app extension to enable a specific task; after users get your extension, they can use it to perform that task in a variety of contexts.

iOS supports app extensions for the following areas, which are known as extension points:

- Share - an extension that enables your app to share content with users on social networks and other sharing services.
- Action - an extension which allows creating custom action buttons in the Action sheet to let users view or transform content originating in a host app.
- Photo Editing - an extension that lets users edit a photo or a video within the Photos app.
- Document Provider - an extension used for allowing other apps to access the documents managed by your app.
- Custom Keyboard - an extension that replaces the system keyboard.

- **Handoff:** Handoff lets users start an activity on one device and seamlessly resume the activity on another device. Provide continuity for users with multiple devices by supporting Handoff in your apps and websites. For example, a user who is browsing a long article in Safari moves to an iOS device that's signed into the same Apple ID, and the same webpage automatically opens in Safari on iOS, with the same scroll position as on the original device. Handoff makes this experience as seamless as possible.

- **Document Picker:** The document picker feature lets users select documents from outside your app's sandbox. These include documents stored in iCloud Drive and documents provided by a third-party extension. Users can open these documents directly, editing them in place. This access simplifies sharing documents between apps and enables more complex workflows. For example, users can easily edit a single document using multiple apps.

- **AirDrop:** AirDrop lets users share photos, documents, URLs, and other kinds of data with nearby devices. Support for sending files to other iOS devices using AirDrop is built into the existing UIActivityViewController class. This class displays different options for sharing the content that you specify. If you are not yet using this class, you should consider adding it to your interface.

Cocoa Touch Frame Work:

The following sections describe the frameworks of the Cocoa Touch layer and the services they offer.

Address Book UI Framework:

- Address Book UI is an iOS framework for displaying, selecting, editing, and creating contacts in a user's Address Book. Similar to the Message UI framework, Address Book UI contains a number of controllers that can be presented modally, to provide common system functionality in a uniform interface.
- To use the framework, add both AddressBook.framework and AddressBookUI.framework to our project,under the "Link Binary With Libraries" phase.
- The Address Book technology for iOS provides a way to store people's contact information and other personal information in a centralized database, and to share this information between applications. The technology has several parts:
- The Address Book framework provides access to the contact information.
- The Address Book UI framework provides the user interface to display the information.
- The Address Book database stores the information.
- The Contacts application provides a way for users to access their contact information.

EventKit UI Framework:

- To work with reminder and calendar events, we need to link against **EventKit**. We will also need a persistent store to save reminder items. Conveniently, EventKit provides this : **EKEventStore**. An **EKEventStore** allows us to fetch, create, edit, and delete events from a user's Calendar database.
- Both reminders and calendar data are stored in the Calendar database. Ideally, we will have only one event store for your entire app, and you will instantiate it once. That's because an **EKEventStore** object requires a relatively large amount of time to initialize and release.

- The Event Kit UI framework provides the classes needed to create, edit, and display events using a view controller. It provides several configurable view controller classes.

GameKit Framework:

Game Kit provides three separate pieces of functionality:

- Game Center offers a centralized game service that connects players to each other. Game Center implements many different features:
- Friends allow players to create anonymous online personas. Users connect to Game Center and interact with other players through analias. Players can set status messages as well as mark other players as friends.
- Multiplayer allows your game to create network matches that connect players through Game Center. Players can invite their friends or be connected to anonymous players. Most importantly, players can receive invitations to join a match even when your game is not running. Your game is running on each device and the instances of your game exchange match and voice data with each other.

2.Media Layer

- The Media layer contains the graphics, audio, and video technologies we use to implement multimedia experiences in our apps. The technologies in this layer make it easy for us to build apps that look and sound great.
- The Media layer is distributed into several components that are specifically classified for graphical, audio or visual support frameworks. The graphical frameworks allow developers to create apps that provide graphical interfaces, animations, image input/output (I/O) readability and access to native visual elements of the device. The audio framework enables the playing, recording and integrating of audio within developed apps.

Graphic Technology:

- A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers, hand-held devices and other appliances. This interface uses icons, menus and other visual indicator (graphics) representations to display information and related user controls, unlike text-based interfaces, where data and commands are in text. GUIl representations are manipulated by a pointing device such as a mouse, trackball, stylus, or a finger on a touch screen.
- iOS provides built-in support for apps running on either Retina displays or standard-resolution displays. For vector-based drawing, the system frameworks automatically use the extra pixels of a Retina display to improve the crispness of your content. And if you use images in your app, UIKit provides support for loading high-resolution variants of your existing images automatically.

Audio Technologies:

iOS offers a rich set of tools for working with sound in your application. These tools are arranged into frameworks according to the features they provide, as follows.

- Use the *Media Player framework* to play songs, audio books, or audio podcasts from a user's iPod library. For details, see Media Player Framework

Reference, iPod Library Access Programming Guide, and the AddMusic sample code project.

- Use the *AV Foundation framework* to play and record audio using a simple Objective-C interface. For details, see AV Foundation Framework Reference and the avTouch sample code project.

- Use the *Audio Toolbox framework* to play audio with synchronization capabilities, access packets of incoming audio, parse audio streams, convert audio formats, and record audio with access to individual packets. For details, see Audio Toolbox Framework Reference and the SpeakHere sample code project.

- Use the *Audio Unit framework* to connect to and use audio processing plug-ins. For details, see Audio Unit Hosting Guide for iOS.

- Use the *OpenAL framework* to provide positional audio playback in games and other applications. iOS supports OpenAL 1.1. For information on OpenAL, see the OpenAL website, *OpenAL FAQ for iPhone OS*, and the *oalTouch* sample code project.

Video Technologies:

- The iOS video technologies provide support for managing static video content in your app or playing back streaming content from the Internet. For devices with the appropriate recording hardware, we can also record video and incorporate it into our app.

iOS supports many industry-standard video formats and compression standards, including the following:

- H.264 video, up to 1.5 Mbps, 640 by 480 pixels, 30 frames per second, Low-Complexity version of the H.264 Baseline Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats

- H.264 video, up to 768 Kbps, 320 by 240 pixels, 30 frames per second, Baseline Profile up to Level 1.3 with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats

- MPEG-4 video, up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats
- Numerous audio formats, including the ones listed in Audio Technologies

AirPlay

- With AirPlay, we can stream music, photos, and videos to our Apple TV, or stream music to our AirPort Express or AirPlay-enabled speakers. And with AirPlay Mirroring, we can display your iOS screen on our Apple TV.
- AirPlay lets our app stream audio and video content to Apple TV and stream audio content to third-party AirPlay speakers and receivers. AirPlay support is built into numerous frameworks—UIKit framework, Media Player framework, AV Foundation framework, and the Core Audio family of frameworks—so in most cases we do not need to do anything special to support it. Any content we play using these frameworks is automatically made eligible for AirPlay distribution. When the user chooses to play your content using AirPlay, it is routed automatically by the system.
- To extend the content displayed by an iOS device, create a second window object and assign it to any UIScreen objects that are connected to the device through AirPlay. Use this technique when the content we display on the attached screen is different than the content displayed on the iOS device.
- The playback classes of the Media Player framework automatically support AirPlay. We can also display Now Playing content on a connected Apple TV using AirPlay.
- Use the AVPlayer class in AV Foundation to manage our app's audio and video content. This class supports streaming its content via AirPlay when enabled by the user.

Media Layer Framework:

The following data describes the media layer frameworks and the services they offer.

Assets Library Framework:

- The assets framework used to access the user photos and video's stored in the devices managed by the photo application.
- Instances of the Photos framework model classes (PHAsset, PHAssetCollection, andPHCollectionList) represent the entities a user works with in the Photos app: assets (images or videos), collections of assets (such as albums or moments), and lists of collections (such as album folders or moment clusters). These objects, also called *photo entities*, are read-only, immutable, and contain only metadata.
- You work with assets and collections by fetching the photo entities you're interested in and then using those objects to fetch the data you need to work with. To make changes to photo entities, you create change request objects and explicitly commit them to the sharedPHPhotoLibrary object. This architecture makes it easy, safe, and efficient to work with the same assets from multiple threads or multiple apps and app extensions.

AV Foundation Framework:

- AVFoundation is one of several frameworks that we can use to play and create time-based audiovisual media.
- It provides an Objective-C interface we use to work on a detailed level with time-based audiovisual data.
- For example, We can use it to examine, create, edit, or reencode media files. We can also get input streams from devices and manipulate video during realtime capture and playback.
- The primary class that the AV Foundation framework uses to represent media is AVAsset.
- The design of the framework is largely guided by this representation. Understanding its structure will help us to understand how the framework works.
- An AVAsset instance is an aggregated representation of a collection of one or more pieces of media data (audio and video tracks).
- It provides information about the collection as a whole, such as its title, duration, natural presentation size, and so on. AVAsset is not tied to particular data format.

<u>Core Audio:</u>

- Core Audio provides software interfaces for implementing audio features in applications you create for iOS .
- In iOS, Core Audio capabilities include recording, playback, sound effects, positioning, format conversion, and file stream parsing will be done.
- Audio Queue Services provides a straightforward, low overhead way to record and play audio in iOS.
- It is the recommended technology to use for adding basic recording or playback features to our iOS.
- A built-in equalizer and mixer that we can use in our applications
- Automatic access to audio input and output hardware.
- APIs to let us manage the audio aspects of our application in the context of a device that can take phone calls
- Optimizations to extend battery life without impacting audio quality.

<u>CoreAudioKit Framework:</u>

- The CoreAudioKit framework (CoreAudioKit.framework) provides standard views for managing connections between apps that support inter-app audio.
- One view provides a switcher that displays the icons of other connected apps and the other view displays the transport controls that the user can use to manipulate the audio provided by the host app.

<u>Core Graphics Framework:</u>

- The Core Graphics framework is a C-based API that is based on the Quartz advanced drawing engine.
- It provides low-level, lightweight 2D rendering with unmatched output fidelity.
- We use this framework to handle path-based drawing, transformations, color management, offscreen rendering, patterns, gradients and shadings,

image data management, image creation, masking, and PDF document creation, display, and parsing.

- Quartz 2D is an advanced, two-dimensional drawing engine available for iOS application development.
- Quartz 2D provides low-level, lightweight 2D rendering with unmatched output fidelity regardless of display or printing device.
- The Quartz 2D API is easy to use and provides access to powerful features such as transparency layers, path-based drawing, offscreen rendering, advanced color management, anti-aliased rendering, and PDF document creation, display, and parsing.

Core Image Framework:

- Core Image is an image processing and analysis technology designed to provide near real-time processing for still and video images. In iOS and OS X we can use Core Image classes to.
- Process images using the many built-in image filters.
- Chain together filters and then archive them for later use.
- Detect features (such as faces and eyes) in still and video images, and track faces in video images.
- Analyze images to get a set of auto adjustment filters.
- Create custom filters for use in our app.

2. List and explain the Application Life Cycle Methods in iOS

Most state transitions are accompanied by a corresponding call to the methods of your app delegate object. These methods are your chance to respond to state changes in an appropriate way. These methods are listed below.

1. **application:willFinishLaunchingWithOptions**:
   - This method is our app's first chance to execute code at launch time. Launch time is an important point in an app's life cycle.

- At launch time, the app delegate is responsible for executing any custom code required to initialize your app.
- For example, the app delegate typically creates the app’s initial data structures, registers for any required services, and tweaks the app’s initial user interface based on any available data.

2. **<u>application:didFinishLaunchingWithOptions:</u>**—
    - This method allows you to perform any final initialization before our app is displayed to the user.

3. **<u>applicationDidBecomeActive:</u>**—
    - Lets our app know that it is about to become the foreground app. Use this method for any last minute preparation
    - This method is called to let your app know that it moved from the inactive to active state.
    - This can occur because your app was launched by the user or the system.
    - Apps can also return to the active state if the user chooses to ignore an interruption (such as an incoming phone call or SMS message) that sent the app temporarily to the inactive state.
4. **<u>applicationWillResignActive:</u>**—
    - Lets us know that our app is transitioning away from being the foreground app.
    - Use this method to put our app into a quiescent state
    - This method is called to let our app know that it is about to move from the active to inactive state.

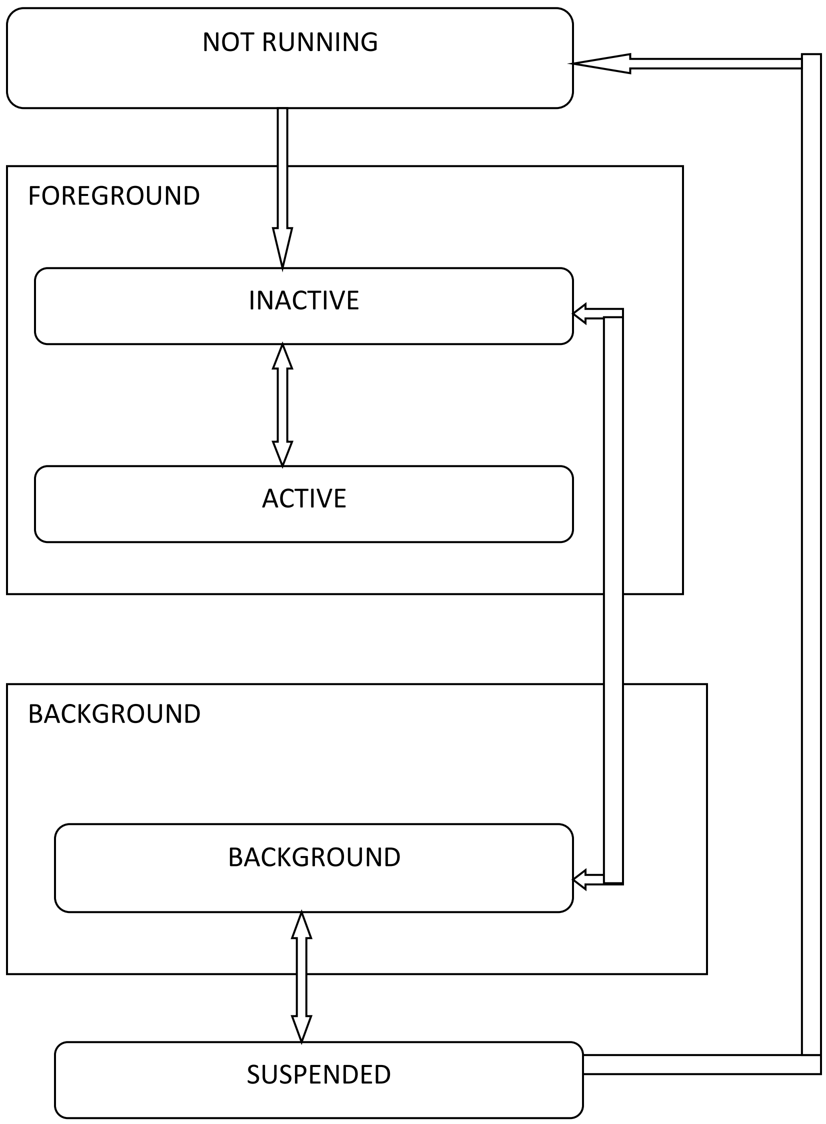
- This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the app and it begins the transition to the background state.
- An app in the inactive state continues to run but does not dispatch incoming events to responders.
- We should use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame rates. Games should use this method to pause the game.

5. **applicationDidEnterBackground:**—
    - Lets us know that our app is now running in the background and may be suspended at any time.
6. **applicationWillEnterForeground**:—
    - Lets us know that our app is moving out of the background and back into the foreground, but that it is not yet active.
7. **applicationWillTerminate**:—
    - Lets us know that our app is being terminated.
    - This method is not called if our app is suspended.

3. Explain Application Life Cycle with an example

- At any given moment, our app is in one of the states .
- The system moves our app from state to state in response to actions happening throughout the system.

- For example, when the user presses the Home button, a phone call comes in, or any of several other interruptions occurs, the currently running apps change state in response.



An iOS app have different states. The different states are listed below.

Not running:

The app has not been launched or was running but was terminated by the system.

Inactive:

The app is running in the foreground but is currently not receiving events. (It may be executing other code though.) An app usually stays in this state only briefly as it transitions to a different state.

Active:

The app is running in the foreground and is receiving events. This is the normal mode for foreground apps.

Background:

The app is in the background and executing code. Most apps enter this state briefly on their way to being suspended. However, an app that requests extra execution time may remain in this state for a period of time. In addition, an app being launched directly into the background enters this state instead of the inactive state. For information about how to execute code while in the background.

Suspended:

The app is in the background but is not executing code. The system moves apps to this state automatically and does not notify them before doing so. While suspended, an app remains in memory but does not execute any code.

When a low-memory condition occurs, the system may purge suspended apps without notice to make more space for the foreground app.

4. What is application Life Cycle?

Application life cycle management is the continuous process of managing the life of an application. Although some people equalte application lifecycle with the software development lifecycle, it comprises much more. In fact, an application’s lifecycle includes entire time during which an organization spend memory on it, from the initial idea of the end of the applications life.