Life cycle

• Life cycle 은 App Lifecycle(앱 생명주기) 과 ViewController Lifecycle(뷰 컨트롤러 생명주기) 로 나뉘어 진다.

•앱을 실행하고 홈 버튼을 눌렀을 경우 background 상태, 화면이 올라와 있는 상태인 foreground상태 등을 정의 한 것

•App 의 life cycle은 앱 실행부터 완전히 종료 될 때 까지 약 9단계로 구분된다.

1. App entry point main() 호출

2. main() 안에서 iOS entry point UIApplicationMain() 호출,  UIApplication 객체 생성

3. UIApplication 객체는 info.plist 파일을 바탕으로 앱에 필요한 데이터와 객체를 로드

4. 커스텀 코드 처리를 위한 AppDelegate객체생성,  UIApplication 객체와 연결

5. 주요 객체 생성, 이벤트 루프 생성 등 실행 준비. 끝나기 직전에 application(\_:willFinishLaunchingWithOptions:) 호출

6. 준비가 끝나고 앱이 실행되기 직전에 application(\_:didFinishLaunchingWithOptions:) 호출

7. Main run loop 실행. 사용자로부터 받은 event를 event queue를 통해 순서대로 처리. 이벤트 핸들(@IBOutlet, @IBAction 등)에 의해 커스텀 코드와 연결되어 실행시킴

8. App을 더 이상 사용하지 않으면 iOS system에 terminate 메시지 전달, applicationWillTerminate(\_:) 호출.

9. App 종료

- 과정 살피기

스크린샷이(가) 표시된 사진

자동 생성된 설명

UIApplication

• UIApplication 객체는 info.plist 파일을 바탕으로 앱에 필요한 데이터와 객체를 로드한다

-> UIApplication 객체는 위의 사진에서 event loop를 관리하고 AppDelegate에 중요 알림을 전달하여 custom code를 처리 하도록 한다.

AppDelegate

• 커스텀 코드 처리를 위한 AppDelegate객체생성, UIApplication 객체와 연결 ->AppDelegate는 custom code를 처리하는 객체. UIApplication을 상속받아 부여 받은 권한으로 app초기화, 상태변화, event등을 관리함

• AppDelegate는 app 하나당 하나의 인스턴스를 갖도록 보장한다 -> 인스턴스란? 같은 클래스에 속하는 개개의 객체로, 하나의 클래스에서 생성된 객체를 말함

• UIApplicationDelegate protocol을 구현하여 app의 상태 변화마다 호출되는 method에 custom code를 작성하여 실행

Delegate Call

• App이 상태변화가 일어나면 AppDelegate에 구현된 UIApplication Delegare method들이 호출 된다

• App의 상태 변화에 따라 의도한 custom code를 실행시킬 수 있다.

• application(\_:willFinishLaunchingWithOptions:)

-> App 실행을 위한 초기화가 완료되기 직전에 호출. 최초로 실행할 코드 작성

• application(\_:didFinishLaunchingWithOptions:)

-> App이 실행되고 사용자에게 보여지기 직전에 호출 최종 초기화 코드 작성

• applicationDidBecomeActive(\_:)

-> App이 Not Running상태에서 Foreground로 진입할 때 호출

• applicationWillResignActive(\_:)

-> App이 Foreground상태에서 다른 상태로 진입 할 때 호출

• applicationDidEnterBackground(\_:)

-> App이 Background 상태에 진입했을 때 호출 특별한 처리가 없다면 suspended 전환

Suspended 상태로 진입하기 전 중요한 데이터를 저장하는 등 종료 전의 준비 작업 진행

• applicationWillEnterForeground(\_:)

-> App이 Background에서 Foreground로 돌아오기 직전 호출

• applicationWillTerminate(\_:)

-> App이 종료되기 직전에 호출 app이 종료될 것임을 알려준다.

° 다음 경우에는 호출 하지 않는다

1. 메모리 확보를 위해 suspended 상태에서 app종료 시
2. 사용자가 multitasking UI를 통해 종료 시
3. 오류로 인해 app이 종료 될 때
4. 디바이스를 재부팅 할 때

App Execution States

:시스템에서 발생하는 동작에 의해 가지는 5가지 상태

스크린샷, 시계이(가) 표시된 사진

자동 생성된 설명

• Not Running

- App이 실행되지 않았거나 완전히 종료된 상태

• Inactive (Foreground)

- App이 foreground 상태에 있지만 이벤트를 받지 않음

• Active (Foreground)

- App이 foreground 상태에 있고 이벤트를 받음

(평소의 app이 머무는 단계)

• Background (Background)

- App이 background 상태에서 다른 코드를 실행

Suspended상태로 넘어가기 전 잠시 머무르지만 추가적인 코드 수행이 필요하면 좀 더 머무른다.

• Suspended

- App이 background 상태에서 메모리만 올려져있고 코드수행은 하지 않음

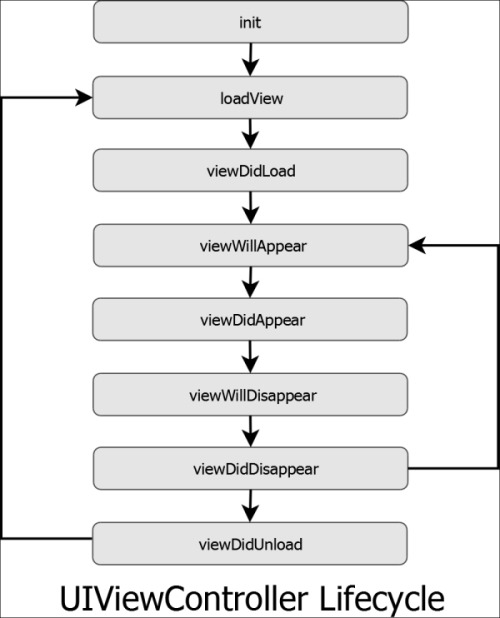
Background 에서 추가적인 작업이 없다면 suspended상태로 진입

App Life Cycle

스크린샷이(가) 표시된 사진

자동 생성된 설명

View Controller Lifecycle



• viewDidLoad

- viewDidLoad 메소드는 뷰의 로딩이 완료되었을 때 시스템에 의해 자동으로 호출된다.

• viewWillAppear

- viewWillAppear 은 뷰가 나타나기 직전에 호출된다.

• viewDidAppear

- viewDidAppear 은 뷰가 나타난 직후에 호출된다는 점에서 viewWillAppear과 차이가 있다.

- viewDidAppear은 화면에 적용될 에니메이션을 그려준다.

• viewWillDisappear

- viewWillDisappear 은 뷰가 사라지기 직전에 호출

• viewDidDisappear

- viewDidDisappear 은 뷰가 사라졌음을 컨트롤러에 알려줌