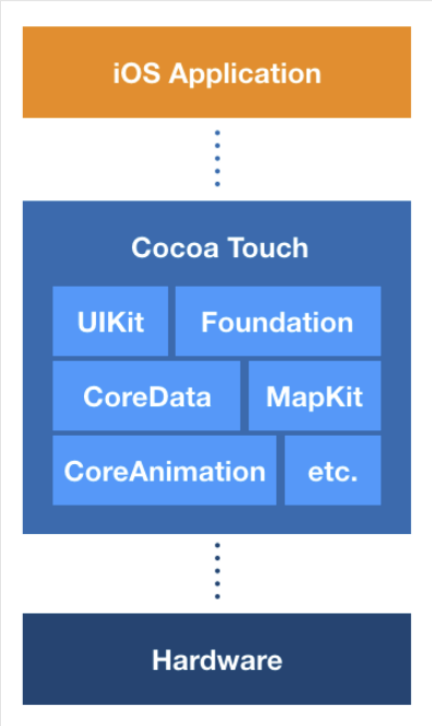
Foundation, UIKit, Cocoa Touch

Cocoa Framework, Touch



Cocoa 이름의 유래?

Java 언어의 이름이 커피산지에서 따온 것은 널리 알려진 사실이다. Cocoa라는 이름은 개발자들이 우리는 어린이를 위한 Java를 만들자는 뜻으로 코코아로 지었다고 한다.

Cocoa Framework ?

프레임워크는 앱의 뼈대를 만들어두는 것이고, 이런 프레임워크를 여러개 모아서 더욱 큰 프레임워크를 구성한 것이 애플의 **코코아 프레임워크**라고 볼 수 있다.

Cocoa Touch ?

터치와 관련된 디바이스의 애플리케이션을 개발할 때 사용할 도구가 **코코아 터치 프레임워크**이다.

iOS 개발을 할 때는 이 코코아 터치 프레임워크를 사용하게 된다. 이 프레임워크에 iOS 개발에서 굉장히 많이 마주치게되는 UIKit과 Foundation 프레임워크가 포함되어있다.

사용하는 방법은?

import로 불러온다.

UIKit Framework

https://melod-it.gitbook.io/sagwa/app-frameworks/uikit

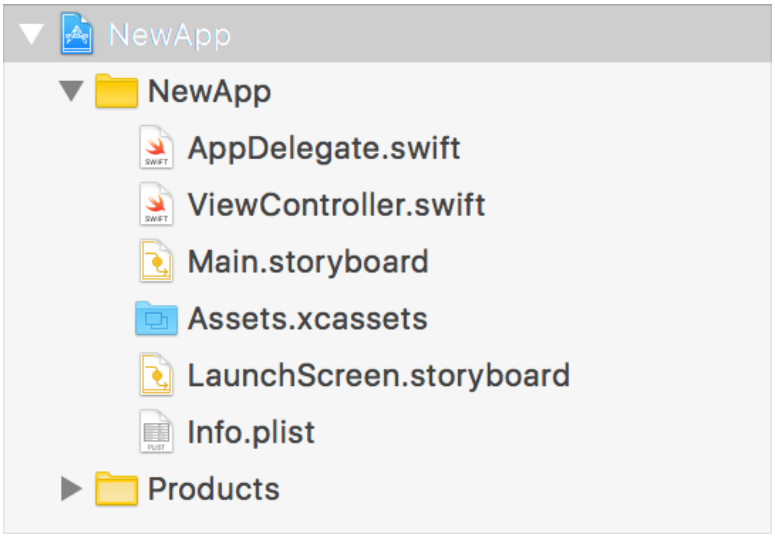
정의

UIKit은 iOS 애플리케이션의 사용자 인터페이스를 구현하고 이벤트를 관리하는 프레임워크입니다.

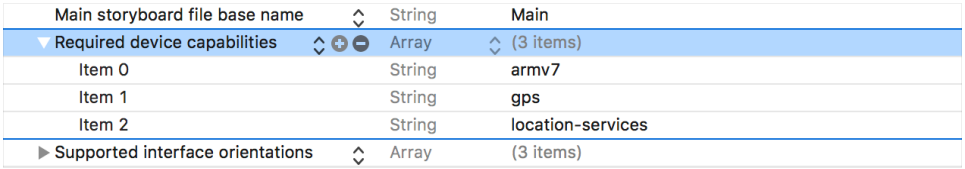
개요

템플릿 프로젝트

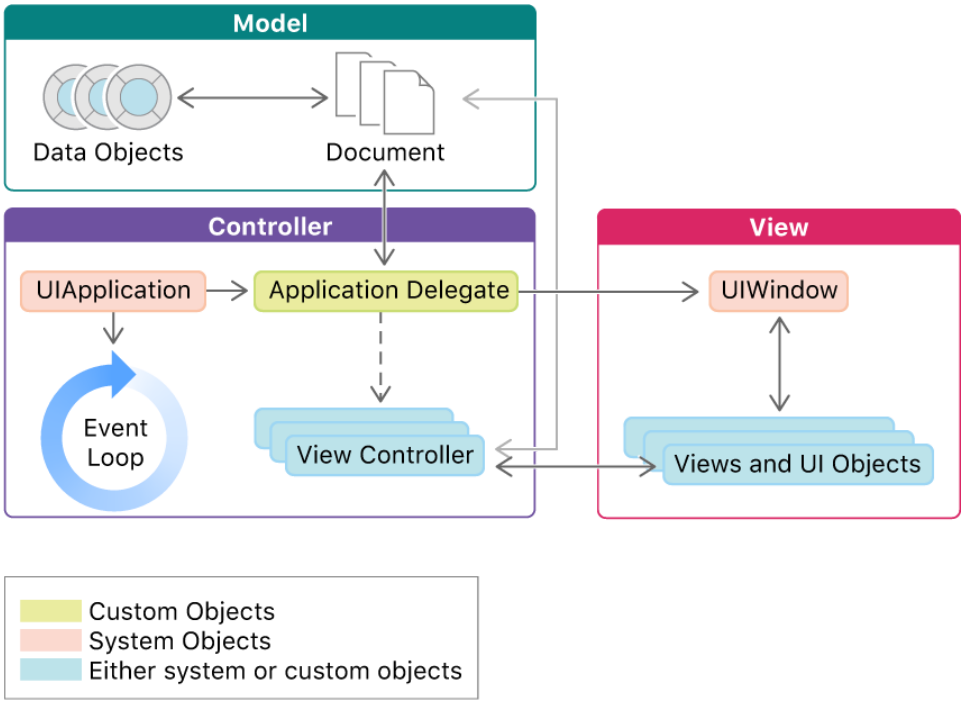
- 리소스 - 이미지, asset, 스토리보드 파일, 문자열 파일 및 애플리케이션 메타데이터



Info.plist



UIKit 앱의 코드 구조



앱 라이프 사이클 관리하기

생명주기 ?

생명주기라는 것은 **앱의 최초 실행부터 앱이 완전히 종료되기까지 앱이 가지는 상태와 그 상태들 사이의 전이**를 뜻합니다.

앱의 상태는 앱이 현재 어떠한 것을 할 수 있는가를 결정합니다.

앱의 생명주기란?

app의 생명 주기는 App의 실행 / 종료 및 App이 Foreground / Background 상태에 있을 때,

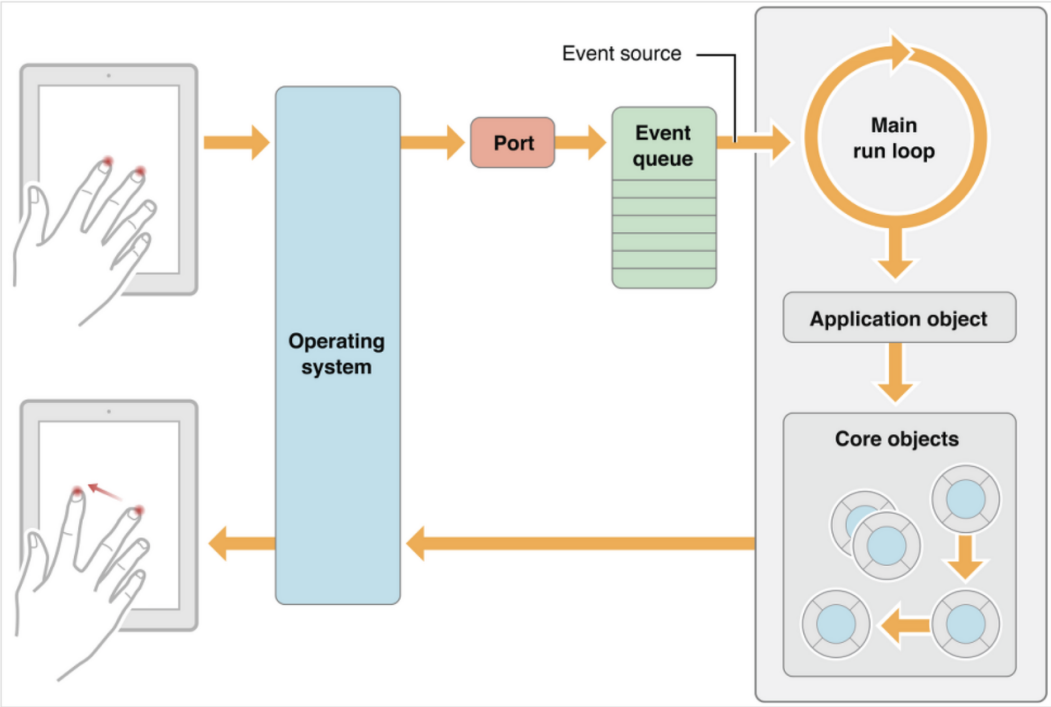
시스템이 발생시키는 Event에 의해 App의 상태가 전환되는 일련의 과정을 뜻합니다.

- Foreground : 앱이 화면에 올라와있는 상태
- Background : 앱이 화면상에서 보여지지 않는 상태

Main Run Loop

Main Run Loop는 모든 사용자가 발생시키는 이벤트에 따라 처리하는 프로세스입니다.

이벤트가 발생했을때 , UIKit에 의해 설정된 Port를 통해 내부의 Event queue에 이벤트를 담아 놓고, 담겨있는 이벤트를 Main Run Loop에서 하나하나씩 실행합니다.



UIApplication 객체는 앱이 실행되거나 이벤트 처리 혹은 View기반의 인터페이스에서 업데이트가 발생할 때 setup 됩니다.

이름에서 볼 수 있듯이 Main Run Loop는 앱의 **Main Thread를 실행** (앱을 돌면서 모든 이벤트를 처리)합니다.

이들은 앱 사용자가 발생하는 이벤트를 순차적으로 처리하기 위해 사용됩니다.

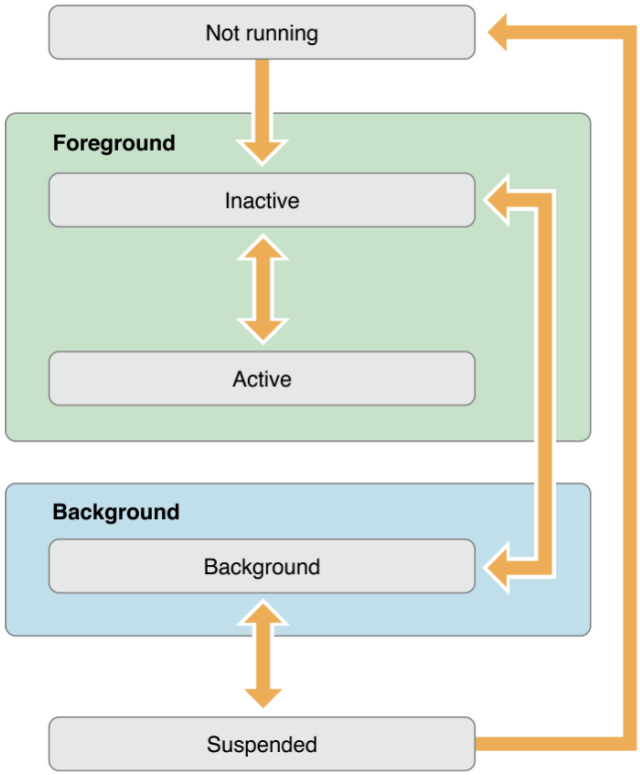
특징

- **UIApplication**은 앱이 생성될 때 생성되고 죽을때까지 **절대 죽지 않습니다**. 즉, 무한 Loop이기 때문에 코드 내에 무한 Loop를 만들면 안 됩니다.
- **Delegate 객체는 어떤 상황이 되기 직전 혹은 직후**에 호출됩니다.
- Application **Delegate**가 **Life Cycle을 전달**받습니다. (Application Delegate를 통해 현재 상태를 알 수 있습니다. UIApplication은 우리가 못 건들기 때문)
- 앱은 **window를 최소 한개이상** 가지고 있습니다.
- **window와 root view controller가 함께 맨 처음 화면**을 띄웁니다.
- iPhone은 1초에 60번, iPad는 120번, 240번 화면을 그립니다. (2019년까지 나온 기기 기준)

이벤트의 처리 과정

1. 유저가 **이벤트**를 일으킨다. (버튼 터치, 화면 스와이프 등 같은 이벤트)
2. 시스템을 통해 이벤트가 **생성**된다.
3. UIKit 프레임워크를 통해 생성된 port로 해당 **이벤트가 앱으로 전달**된다.
4. 이벤트는 앱 내부적으로 Queue의 형태로 정리되고 **Event Queue에 쌓인다**.
5. Event Queue에 있는 이벤트들이 **Main Run Loop에 하나씩 매핑**된다.
6. UIApplication 객체는 이때 가장 먼저 이벤트를 객체로 **어떤 것이 실행되어야 하는지 결정**한다.

앱 상태(App State)



1. Not Running

앱이 실행되지 않았거나, 완전히 종료되어 **동작하지 않는 상태**

2. Foreground

앱이 전면에서 **실행**되고 있는 상태

Inactive

앱이 실행되면서 **foreground에 진입**하지만, 앱 실행 중 미리알림 또는 일정 얼럿이 화면에 덮여서 앱이 실질적으로는 **이벤트를 받지 못하는 상태**

앱의 상태 전환 과정에서 **잠깐 머무는 단계**

ex) 미리알림 등의 alert가 올라와 앱이 실질적으로 이벤트를 받지 못하는 상황

전화가 왔을 때, 시스템 메시지가 떴을 때 (예 : 배터리 부족)

active

앱이 **실행 중**이며, **foreground**에 있고, **이벤트를 받고 있는 상태**

어플리케이션이 **실질적으로 활동**하고 있는 상태

3. Background

앱이 백그라운드에 있으며, **다른 앱으로 전환되었거나 홈 버튼을 눌러 밖으로 나갔을 때의 상태**

- Background 상태로 실행되는 app은 inactive 대신 background 상태로 진입
- **Suspended 상태가 되기 전 잠깐 머무는 상태**(추가 코드 실행이 필요하면 머무는 시간 연장)
- 코드가 실행중이지만 **사용자의 이벤트를 받을 수는 없음** 추가적인 코드 (ex. 파일 다운로드)를 실행하는 동안 머무름
- 홈버튼을 두번 눌러 앱을 다시 열었을 때, 처음부터 재실행되지 않는다면 background 상태에 있다가 올라온 것

ex) background에서 음악을 재생하거나 거리를 추적하고 있는 상황

4. Suspended

앱이 Background 상태에 있지만, **아무 코드도 실행하지 않은 상태**입니다.

백그라운드에서 특별한 작업이 없을 경우 Suspended 상태가 됩니다.

이 상태에서 앱은 메모리상에 올라가있지만 **아무 일도 하지 않기 때문에 배터리를 사용하지 않습니다.**

또한 OS에 의해 메모리 부족 현상이 발생하면 이 상태의 앱은 메모리에서 없어질 수 있으며, 이는 따로 알림을 하거나 하진 않습니다.

참고

https://coding-sojin2.tistory.com/196

Foundation Framework

https://melod-it.gitbook.io/sagwa/app-frameworks/foundation

Foundation은 **iOS 애플리케이션의 운영체제 서비스와 기본 기능을 포함하는 프레임워크**입니다.