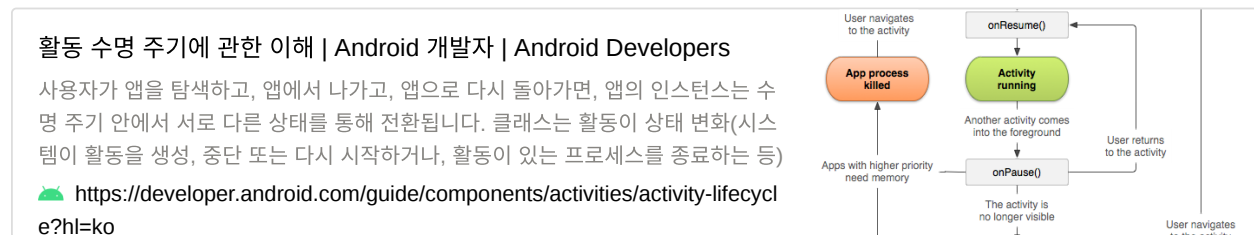




LifeCycle

Activity LifeCycle




하나의 activity는 여러 상태를 통과하면서 시작하고 끝나는데 여러 callback method를 통해 상태 변화가 일어나는데 어떤 상황에서 상태가 변화하는지를 알아야지 정확한 순간에 시작, 일시중지, 종료를 결정해서 오류, 데이터 손실 없이 activity를 조작할 수 있다.

1. onCreate(): activity가 생성될 때 불려지는 메소드로 초기화를 담당한다. 이때 view를 생성하고 activity에 필요한 data를 가져온다. 화면이 만들어지지 않았다.
2. onStart(): onCreate() 메소드에서 생성 준비를 마치면 activity를 본격적으로 시작하는 단계로 view를 만든다. 화면은 만들지만 상호작용은 안된다.
3. onResume(): 본격으로 사용자와 상호 작용을 하는 단계로 이때 여러 activity가 동시에 생성되어 있으면 사용자와 상호작용하는 onResume() 상태에 있는 activity가 activity stack의 가장 상단에 위치하게 된다. 앱의 대부분 기능은 onResume()에 존재한다. 상호작용이 가능하다.
4. onPause(): onResume()상태의 activity가 잠시 멈춰야 할 때 실행된다. 보통 뒤로가기 버튼을 누를 때 실행된다. 이때 activity는 존재하고 보이지만 잠깐 멈출 때 실행되지만 사실 곧바로 onStop()상태로 돌입하면서 activity는 멈춘다. 또는 곧바로 onResume()상태로 돌아가 원래 기능을한다. 따라서 아주 잠시 존재하는 상태이다. 고로 이때 데이터나 변화를 저장하려고 하면 안된다. 화면이 가려질 때 아주 잠깐 호출, 뭐를 저장할 수는 없지만 잠깐 가려지는 것을 고를 수 있다.

- Use of `onDestroy()` in Android

Thanks for contributing an answer to Stack Overflow! Please be sure to answer the question. Provide details and share your research! Asking for help, clarification, or responding to other answers. Making statements

 <https://stackoverflow.com/questions/13927269/use-of-ondestroy-in-android>

-
- ```
graph TD
 A([Activity launched]) --> B[onCreate()]
 B --> C[onStart()]
 C --> D[onResume()]
 D --> E([Activity running])
 E -- "Another activity comes into the foreground" --> F[onPause()]
 F -- "User returns to the activity" --> D
 F -- "The activity is no longer visible" --> G[onStop()]
 G -- "User navigates to the activity" --> C
 G -- "User navigates to the activity" --> H[onRestart()]
 H --> C
 G -- "The activity is finishing or being destroyed by the system" --> I[onDestroy()]
 I --> J([Activity shut down])
 G -- "Apps with higher priority need memory" --> K([App process killed])
 K -- "User navigates to the activity" --> B
```
- The flowchart illustrates the lifecycle of an Android Activity. It begins with 'Activity launched', followed by the method calls `onCreate()`, `onStart()`, and `onResume()`, leading to the 'Activity running' state. From 'Activity running', the lifecycle can transition to `onPause()` if 'Another activity comes into the foreground'. From `onPause()`, the user can 'return to the activity' to resume `onResume()`, or the activity can become 'no longer visible', leading to `onStop()`. From `onStop()`, the user can 'navigate to the activity' to restart it via `onRestart()` and `onStart()`, or the activity can be 'finishing or being destroyed by the system', leading to `onDestroy()` and 'Activity shut down'. Alternatively, from `onStop()`, the 'App process killed' if 'Apps with higher priority need memory', and the user can 'navigate to the activity' to restart it from `onCreate()`.

### ▷ 언제까지 LifeCycle이 유지되는가

백그라운드: 해당 activity가 activity stack에 존재하되 보이지 않을 때 background에 있다고 한다. (뒤로가기를 누를때)

| kill 가능성 | 프로세스 상태                           | Activity 상태               |
|----------|-----------------------------------|---------------------------|
| 최소       | Foreground (포커스가 있거나 포커스를 가져올 예정) | Created, Started, Resumed |
| 중간       | Background (포커스 상실)               | Paused                    |
| 최대       | Background(보이지 않음)Empty           | StoppedDestroyed          |

### ▷임시 UI 상태 저장 및 복원

사용자는 일시적으로 앱 밖으로 나가 다른 일을 하고 다시 앱에 진입할 때 UI상태가 그대로 유지되기를 희망한다. Activity가 소멸된 뒤 앱에 다시 재진입할 때 복원을 하기 위해 ViewModel, onSaveInstanceState() 또는 로컬 저장소에 UI 상태를 임시저장해야 한다.

간단한 UI의 경우 onSaveInstanceState()만으로 UI를 복구할 수 있지만 대부분의 경우 ViewModel, onSaveInstanceState()을 모두 사용해야 한다.

#### ▼ ViewModel



<https://developer.android.com/topic/libraries/architecture/saving-states?hl=ko>

#### ▼ onSaveInstanceState()를 사용하여 간단하고 가벼운 UI 상태 저장

1. Activity가 onStop되려고 하면 onSaveInstanceState() 메서드가 callback된다. 이 때 activity를 instance state Bundle에 임시저장할 수 있다. 여기에는 widget의 중간 상태와 scroll 위치 등이 저장된다. 특정 상태를 추가로 저장하고 싶으면 onSaveInstanceState()를 override해서 Bundle 객체에 key-value 쌍을 저장해야 한다.

```
override fun onSaveInstanceState(outState: Bundle?) {
 // Save the user's current game state
 outState?.run {
 putInt(STATE_SCORE, currentScore)
 putInt(STATE_LEVEL, currentLevel)
 }

 // Always call the superclass so it can save the view hierarchy state
 super.onSaveInstanceState(outState)
}

companion object {
 val STATE_SCORE = "playerScore"
```

```
val STATE_LEVEL = "playerLevel"
}
```

2. Activity가 파괴 후 다시 시작될 때 onCreate()와 onRestoreInstanceState()가 callback되면서 임시 상태를 저장하고 있는 Bundle을 가져오게 된다.

```
override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState) // Always call the superclass first

 // Check whether we're recreating a previously destroyed instance
 if (savedInstanceState != null) {
 with(savedInstanceState) {
 // Restore value of members from saved state
 currentScore = getInt(STATE_SCORE)
 currentLevel = getInt(STATE_LEVEL)
 }
 } else {
 // Probably initialize members with default values for a new instance
 }
 // ...
}
```

```
override fun onRestoreInstanceState(savedInstanceState: Bundle?) {
 // Always call the superclass so it can restore the view hierarchy
 super.onRestoreInstanceState(savedInstanceState)

 // Restore state members from saved instance
 savedInstanceState?.run {
 currentScore = getInt(STATE_SCORE)
 currentLevel = getInt(STATE_LEVEL)
 }
}
```

▷ 두개의 activity 사이 전환이 일어날 때

1. 활동 A의 onPause() 메서드가 실행됩니다.
2. 활동 B의 onCreate(), onStart() 및 onResume() 메서드가 순차적으로 실행됩니다. (이제 사용자 포커스는 활동 B에 있습니다.)
3. 그런 다음, 활동 A가 더 이상 화면에 표시되지 않는 경우 이 활동의 onStop() 메서드가 실행됩니다.

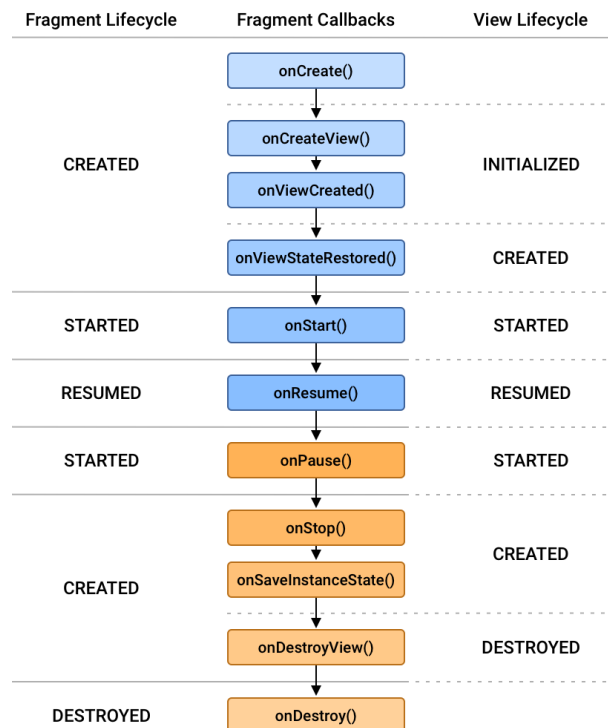
## Fragment Lifecycle

## Fragment lifecycle | Android Developers

Each instance has its own lifecycle. When a user navigates and interacts with your app, your fragments transition through various states in their lifecycle as they are added, removed, and enter or exit the screen. To manage lifecycle,

 <https://developer.android.com/guide/fragments/lifecycle>

Developers 



Fragment의 view는 Fragment와는 독립적인 lifecycle을 가진다.

- Fragment CREATED and View INITIALIZED
- Fragment and View CREATED
- Fragment and View STARTED
- Fragment and View RESUMED
- Fragment and View STARTED
- Fragment and View CREATED

- Fragment CREATED and View DESTROYED
- Fragment DESTROYED

## Lifecycle Aware Component

### Handling Lifecycles with Lifecycle-Aware Components | Android Developers

Lifecycle-aware components perform actions in response to a change in the lifecycle status of another component, such as activities and fragments. These components help you produce better-organized, and often lighter-weight code, that is easier to

 <https://developer.android.com/topic/libraries/architecture/lifecycle>

Developers 