

안드로이드 SDK 개발 플랫폼 활용하기

액티비티 라이프 사이클

- 액티비티는 안드로이드 앱을 구성하는 기본단위이며, 각 화면을 의미
- **액티비티의 4가지 상태**
 1. Active(활동적이거나 실행 중)
 - 화면 전면에 있을 때
 - 현재 태스크의 액티비티 스택에 맨 위에 위치
 - 사용자의 액션에 포커스를 둔 액티비티
 2. Pause(일지정지 됨)
 - 포커스를 잃었지만 사용자에게 보여짐
 - 다른 액티비티가 위에 놓여져 있고 투명하거나 화면 전체를 덮진 않는다. 그래서 일시 정지 된 몇 액티비티는 보여질 수도 있음
 - 모든 상태와 멤버 정보를 유지하고 윈도우 매니저에 속해 있지만 극도로 메모리가 낮을 때 시스템에 의해 꺼질 수 있음

● 액티비티의 4가지 상태

3. Stop(정지 됨)

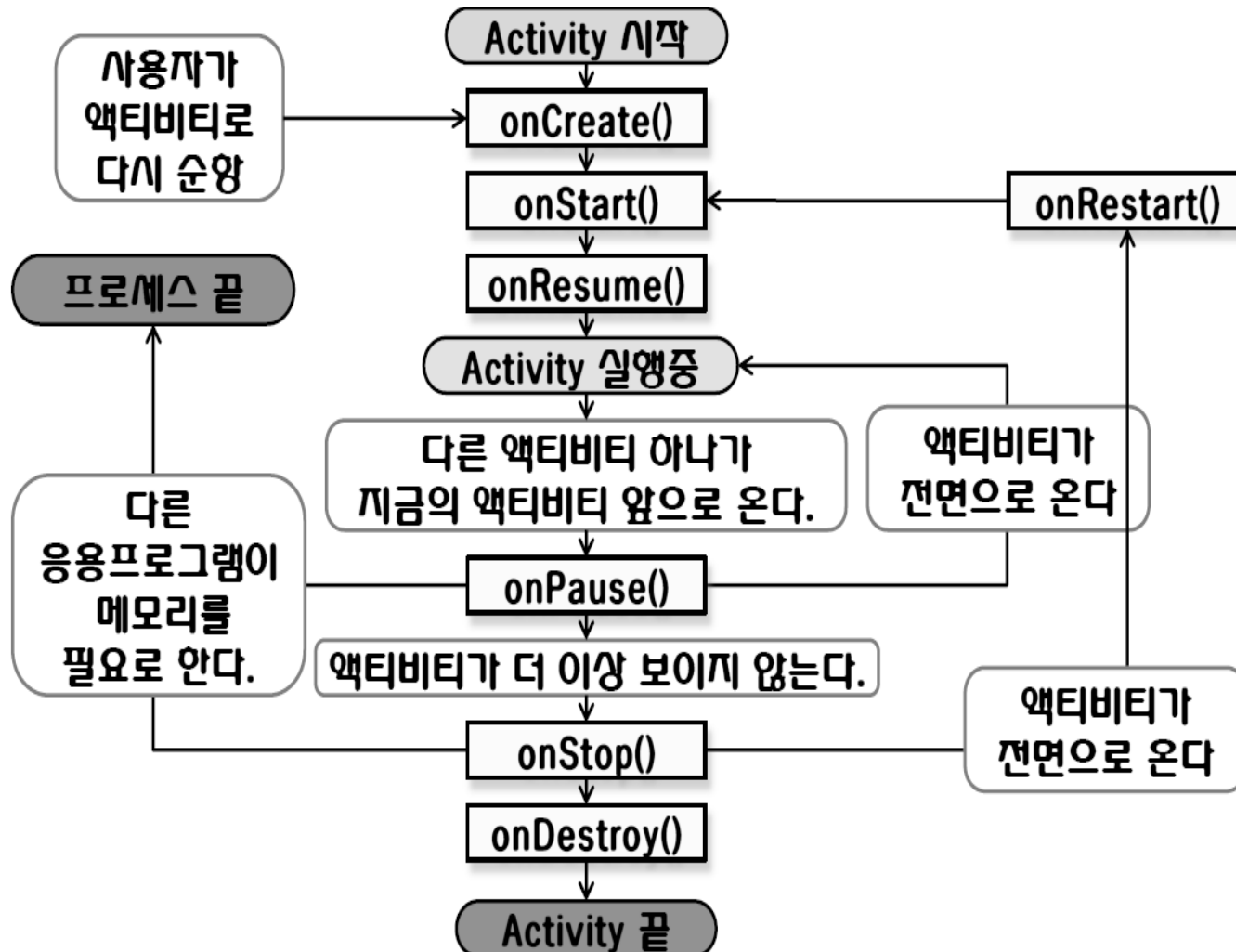
- 완전히 다른 액티비티에 가려졌을 때 모든 상태와 멤버 정보는 유지
- 하지만 사용자에게 더 이상 보여지지 않아 창이 가려지고 메모리가 다른 곳에 필요할 때 꺼짐

4. Inactive(실행되지 않은 상태)

- 일반적으로 앱을 설치하고 나면 실행되지 않는 상태로 존재
- 또는 앱을 설정에서 강제종료 시키면 실행되지 않은 상태가 됨

- 액티비티의 상태 변화에 따라 다음의 메소드 호출
 1. `void onCreate(Bundle savedInstanceState)`
 - 화면이 물리적인 메모리 공간에 생성되는 상태로 화면에는 보이지 않는다.
 2. `void onStart()`
 - 화면의 동작이 실행되는 상태로 액티비티가 만들어지고 바로 호출된다.
 3. `void onRestart()`
 - 앱이 Stop 되었다가 다시 화면에 나타나는 경우 호출 된다.
 4. `void onResume()`
 - 앱이 일시 정지 되었다가 다시 실행되는 경우 호출 된다.
 5. `void onPause()`
 - 앱이 정지되기 전에 가장 먼저 일시 정지 상태를 거친다.
 6. `void onStop()`
 - 앱이 다른 앱에 의해 가려지는 경우 정지 상태로 넘어 간다.
 7. `void onDestroy()`
 - 화면이 종료되는 경우 호출 된다.

- 4개의 상태와 7가지 이벤트의 흐름

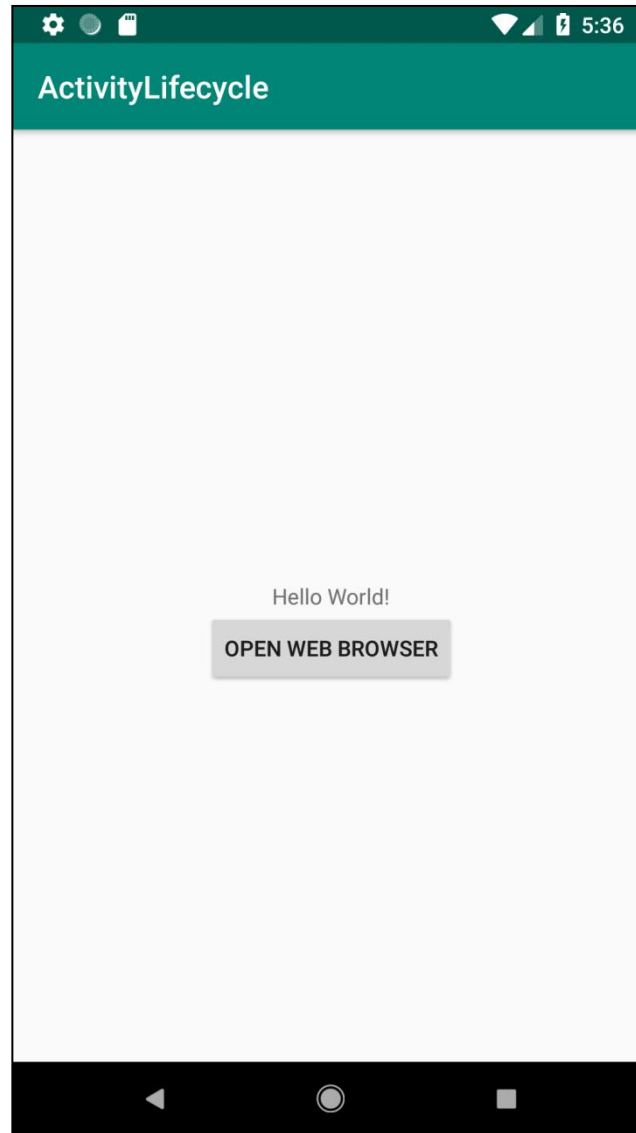


- 액티비티가 겹치는 상황을 만들기 위해 다음과 같이 화면 구성

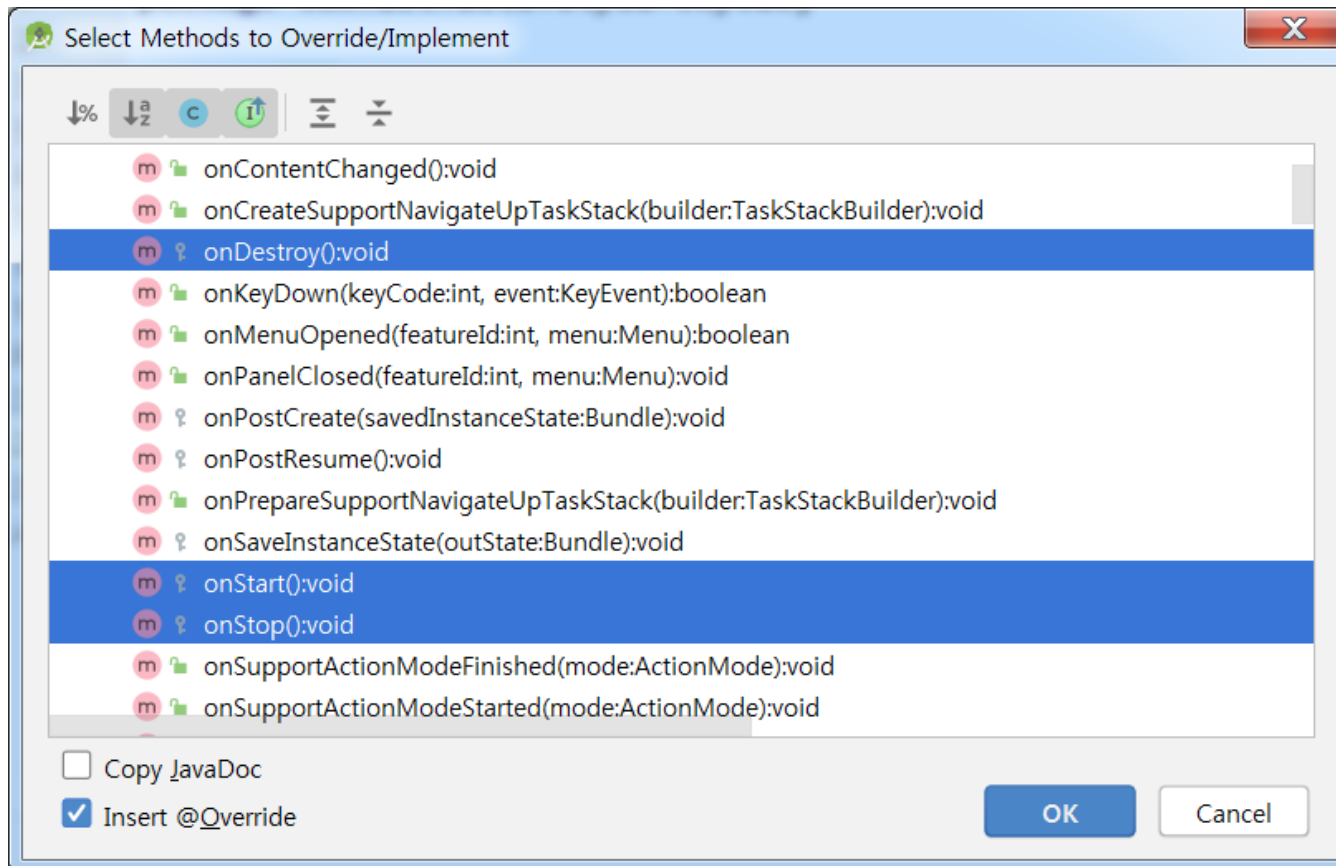
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent" // 부모를 안벗어나게 최대화
    android:layout_height="match_parent" // 부모를 안벗어나게 최대화
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content" // 콘텐츠에 맞게 최소화
        android:layout_height="wrap_content" // 콘텐츠에 맞게 최소화
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent" // 부모를 기준으로 맞춤
        app:layout_constraintLeft_toLeftOf="parent" // 부모를 기준으로 맞춤
        app:layout_constraintRight_toRightOf="parent" // 부모를 기준으로 맞춤
        app:layout_constraintTop_toTopOf="parent" // 부모를 기준으로 맞춤 />
```

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content" // 컨텐츠에 맞게 최소화
    android:layout_height="wrap_content" // 컨텐츠에 맞게 최소화
    android:text="Open Web Browser"
    app:layout_constraintTop_toBottomOf="@id/textView"
    app:layout_constraintLeft_toLeftOf="@id/textView"
    app:layout_constraintRight_toRightOf="@id/textView"
/>
</android.support.constraint.ConstraintLayout>
```



- 오버라이드 하는 방법
 - Ctrl + O
 - 소스코드에 마우스 오른쪽 클릭
 - Generate > Override Methods



- 버튼을 누르면 웹 브라우저를 열어서 기존의 화면이 stop 상태로 변화하는 기능 확인
 - 앞서 배운 로그캣으로 그 상태를 관찰해 보자

```
package com.iot.activitylifecycle;

import android.content.Intent; // 인텐트 활용
import android.net.Uri; // URI 경로 객체 활용
import android.support.v7.app.AppCompatActivity; // 액티비티 부모 클래스
import android.os.Bundle; // 액티비티 생성 번들
import android.util.Log; // 로그 출력 목적
import android.view.View; // 뷰계열 최상위 클래스
import android.widget.Button; // 버튼 뷰 사용

public class MainActivity extends AppCompatActivity { // 메인 화면

    @Override // 부모 메소드 재정의
    protected void onCreate(Bundle savedInstanceState) { // 화면생성 이벤트
        super.onCreate(savedInstanceState); // 부모 생성자 호출
        setContentView(R.layout.activity_main); // 메인 화면 표시
        Button button = (Button)findViewById(R.id.button);
```

```
button.setOnClickListener(new View.OnClickListener() { // 클릭리스터 생성
    @Override // 부모 메소드 재정의
    public void onClick(View v) { // 클릭 이벤트 처리
        Intent intent = new Intent(Intent.ACTION_VIEW,
            Uri.parse("http://www.google.com"));
        startActivity(intent);
    }
});
Log.i("test", "onCreate");
}

@Override // 부모 메소드 재정의
protected void onDestroy() {
    super.onDestroy();
    Log.i("test", "onDestroy");
}

@Override // 부모 메소드 재정의
protected void onStart() {
    super.onStart();
    Log.i("test", "onStart");
}
```

```
@Override // 부모 메소드 재정의
protected void onStop() {
    super.onStop();
    Log.i("test", "onStop");
}
```

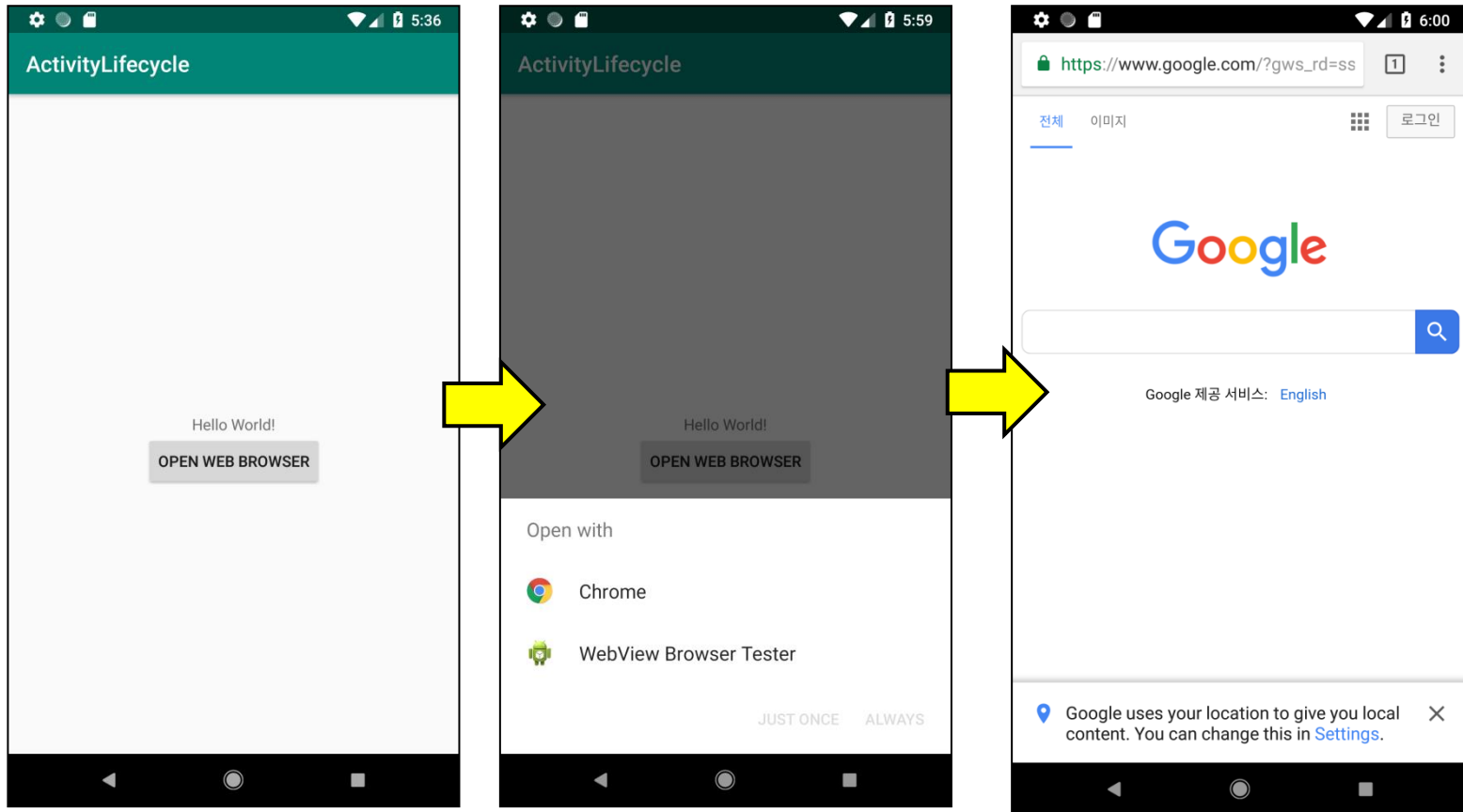
```
@Override // 부모 메소드 재정의
protected void onPause() {
    super.onPause();
    Log.i("test", "onPause");
}
```

```
@Override // 부모 메소드 재정의
protected void onResume() {
    super.onResume();
    Log.i("test", "onResume");
}
```

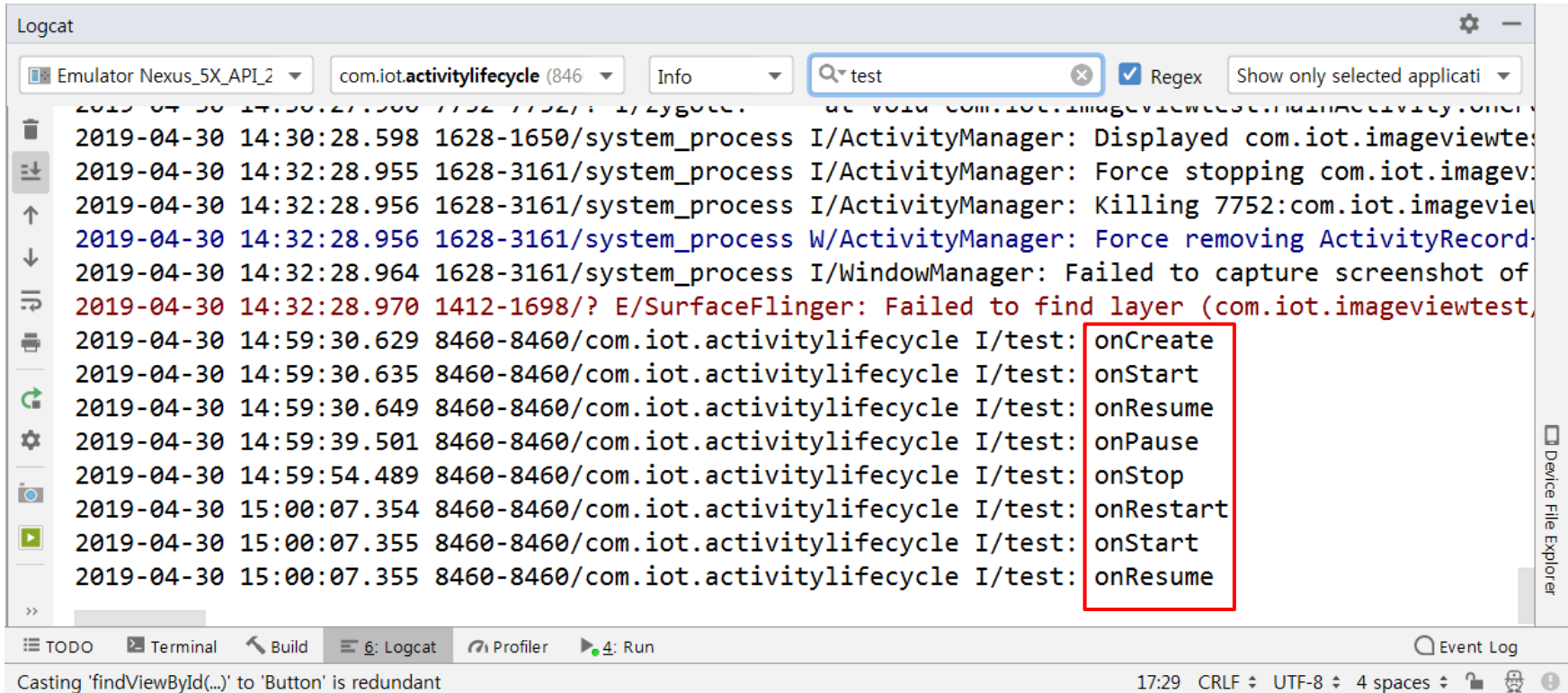
```
@Override // 부모 메소드 재정의
protected void onRestart() {
    super.onRestart();
    Log.i("test", "onRestart");
}
```

```
}
```

- 로그에서 라이프사이클 확인하기
 - 로그켓을 이용하여, 액티비티의 상태변화가 확인되는지 테스트를 해 보자.



- 필터를 이용해 test라는 태그를 가지는 로그 메시지만 필터링 해서 확인



Logcat

Emulator Nexus_5X_API_2 | com.iot.activitylifecycle (846) | Info | test | Regex | Show only selected applicati

```
2019-04-30 14:30:27.500 7752-7752/? I/zygote: ...
2019-04-30 14:30:28.598 1628-1650/system_process I/ActivityManager: Displayed com.iot.imageviewtest
2019-04-30 14:32:28.955 1628-3161/system_process I/ActivityManager: Force stopping com.iot.imageviewtest
2019-04-30 14:32:28.956 1628-3161/system_process I/ActivityManager: Killing 7752:com.iot.imageviewtest
2019-04-30 14:32:28.956 1628-3161/system_process W/ActivityManager: Force removing ActivityRecord{...}
2019-04-30 14:32:28.964 1628-3161/system_process I/WindowManager: Failed to capture screenshot of ...
2019-04-30 14:32:28.970 1412-1698/? E/SurfaceFlinger: Failed to find layer (com.iot.imageviewtest)
2019-04-30 14:59:30.629 8460-8460/com.iot.activitylifecycle I/test: onCreate
2019-04-30 14:59:30.635 8460-8460/com.iot.activitylifecycle I/test: onStart
2019-04-30 14:59:30.649 8460-8460/com.iot.activitylifecycle I/test: onResume
2019-04-30 14:59:39.501 8460-8460/com.iot.activitylifecycle I/test: onPause
2019-04-30 14:59:54.489 8460-8460/com.iot.activitylifecycle I/test: onStop
2019-04-30 15:00:07.354 8460-8460/com.iot.activitylifecycle I/test: onRestart
2019-04-30 15:00:07.355 8460-8460/com.iot.activitylifecycle I/test: onRestart
2019-04-30 15:00:07.355 8460-8460/com.iot.activitylifecycle I/test: onStart
2019-04-30 15:00:07.355 8460-8460/com.iot.activitylifecycle I/test: onResume
```

TODO | Terminal | Build | 6: Logcat | Profiler | 4: Run | Event Log

Casting 'findViewById(...)' to 'Button' is redundant | 17:29 | CRLF | UTF-8 | 4 spaces