

안드로이드 SDK 개발 플랫폼 활용하기

# 쓰레드와 쓰레드 함수

- Thread
  - 임의의 작업들을 병렬로 실행가능하게 하는 자바 객체
  - Thread를 이용하면 멀티태스킹 구현 가능
    - 멀티태스킹이란 타임퀀텀(Time Quantum)이라고 하는 사람이 느낄 수 없는 정도로 짧은 시간마다 CPU를 번갈아 사용하는 것을 뜻함
  - 각 Thread는 CPU로부터 타임 쿼텀을 번갈아 할당받아 할당된 작업을 수행
- 안드로이드는 Single Thread GUI Model이며 아래의 특징을 가짐
  - UI를 그리거나 갱신하는 Thread는 단 한개
  - UI Thread가 멈추면 화면 갱신 불가
  - 5초 이상 응답이 없으면 ANR 팝업 (Application Not Responding)
  - ANR 시 어플리케이션을 강제 종료해야 함

- Thread의 call stack의 특징
  - 실행할 메소드, 인자, local 변수를 위한 별도의 공간을 가짐
  - VM(Virtual Machine)을 통한 작업 수행
  - 각 어플리케이션은 별도의 VM을 할당 받아 동작
  - 같은 VM 내부의 Thread들은 상호작용(interact)이 가능
  - Thread간의 동기화(Synchronization)를 위해 여러 메소드 제공
  - Thread의 동기화 문제
  - 안드로이드를 포함한 타환경에서도 매우 중요하게 고려됨

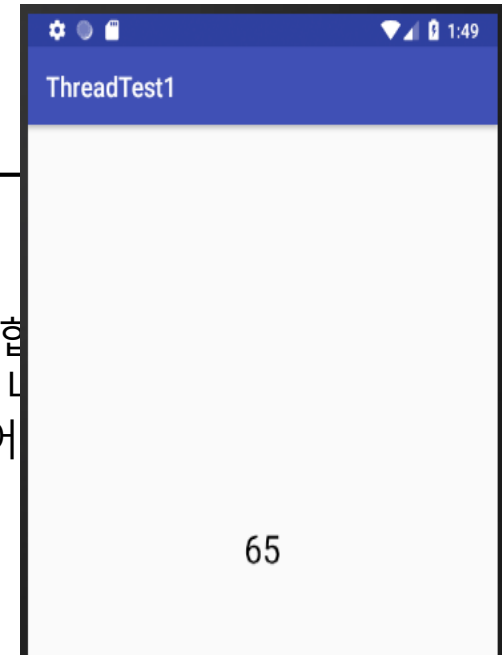
- Thread 객체사용법
  - 1.Thread 클래스를 상속받는 방법
    - ① Thread 클래스 subclassing
    - ② run() 메소드 오버라이딩
  - 2. Runnable인터페이스를 지정해주는 방법
    - ① new 연산자로 새로운Thread 객체 생성
    - ② 생성인자로 Runnable인터페이스
    - ③ Runnable은Thread에서 실행될 logic포함
      - Runnable의 run() 메소드
      - 단 하나의 abstract 메소드 제공
      - 새 work Thread가 실행 할 작업 포함
      - 생성된 새work thread가 시작되면 자동으로 호출

- UI Thread와 work Thread가 동시에 위젯에 접근할 경우 대처 방법
  - ① Thread, Runnable을 이용한 방법
    - 위젯 접근은 UI Thread가 할 수 있도록 요청하는 방식
  - ② AsyncTask 이용한 방법
    - UI Thread가 work Thread에게 UI 위젯 제어를 위임하는 방식
- 동기화 문제를 해결하기 위한 방법
  - 1.Handler를 이용한 메시지 통신
  - 2.Activity.runOnUiThread(Runnable)
    - UI Thread의 메세지큐에작업전달
  - 3.View.post(Runnable)
    - 해당view에 접근가능한경우
  - 4.View.postDelayed(Runnable, long)
    - 일정시간후에 작업처리

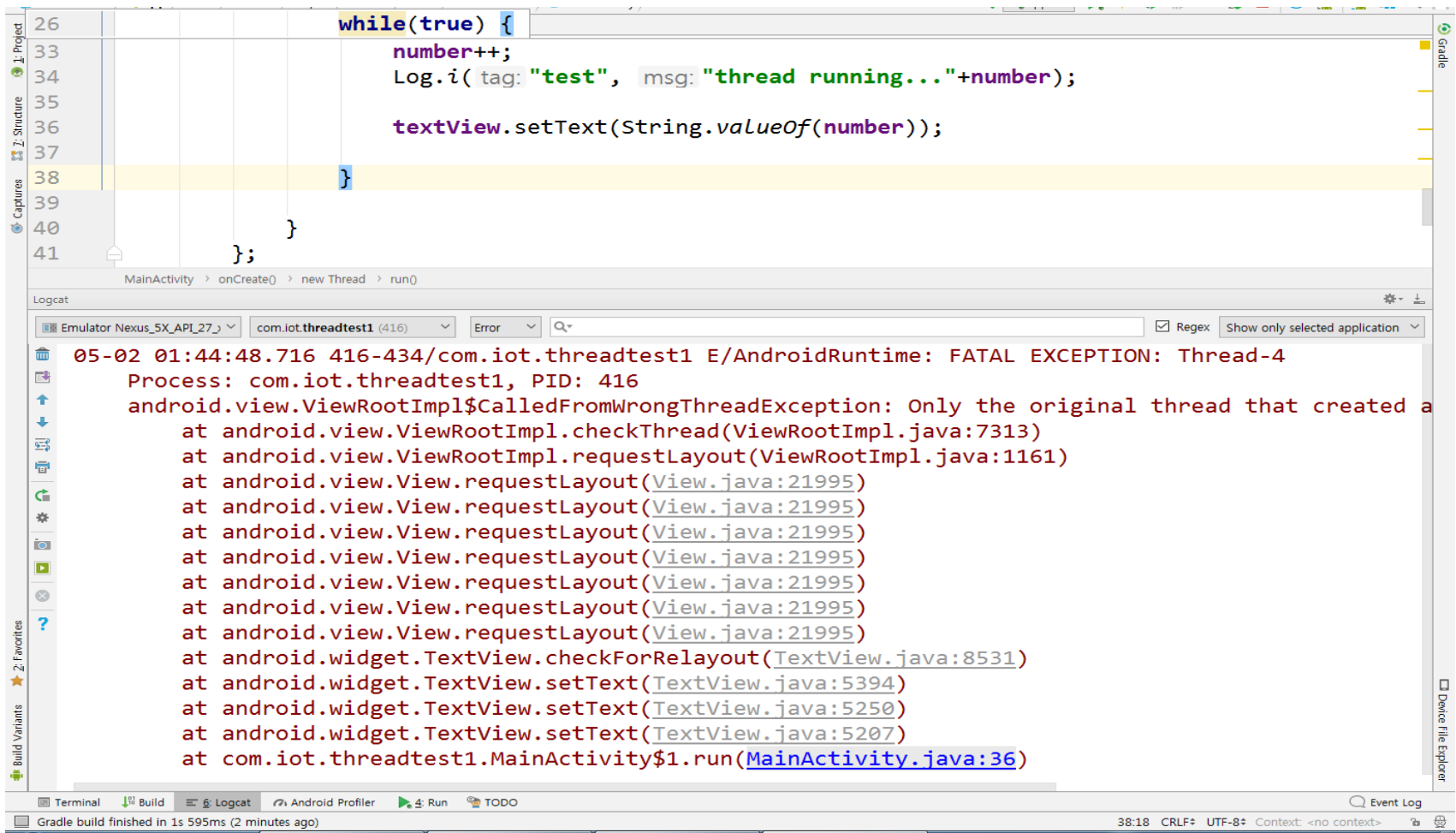
- AsyncTask 사용해 background 작업구현 시 유의사항
  - 1.AsyncTask클래스
    - ① 항상 subclassing하여 사용할 것
    - ② AsyncTask 인스턴스는 항상UI Thread에서 생성할 것
  - 2.AsyncTask:execute(...) 메소드
    - ① 항상 UI Thread내부에서 호출할 것
    - ② 생성된 AsyncTask 인스턴스 별로 꼭 한번만 사용 가능
    - ③ 같은 인스턴스가 또 execute(...)를 실행하면 exception이 발생
    - ④ AsyncTask:cancel(...) 로 취소된 AsyncTask 인스턴스도 동일
    - ⑤ 매번 new 로 AsyncTask인스턴스를 새로 생성할 것
  - 3.AsyncTask의 callback 함수를 직접 호출하면 안되므로 주의
    - AsyncTask의 콜백함수는 onPreExecute(), doInBackground(...), onProgressUpdate(...), onPostExecute(...)등이 있는데 이런 메소드는 callback으로만 사용

- 스레드에서 화면에 접근할 경우 예외가 발생할 경우 화면에 비동기적으로 접근하거나 `runOnUiThread()`를 이용
- 1초에 1씩 숫자를 증가시키는 역할을 하는 예제

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="안드로이드 APK 파일의 스키마 경로는 생략함"
    android:layout_width="match_parent" // 부모를 안벗어나
    android:layout_height="match_parent" // 부모를 안벗어나
    android:orientation="vertical"
    android:gravity="center"
    >
    <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!"
        android:textSize="28dp"
        android:textColor="#000000"
    />
</LinearLayout>
```



- 만약 Thread에서 화면에 있는 TextView에 직접 접근하려고 하면 UI Thread가 UI를 생성한 Original Thread의 View에 접근할 수 없다는 예외 발생
- 안드로이드는 Single UI Thread 모델이며, Background Thread는 화면에 접근하려면 비동기적인 방식을 이용해야 함. 예를 들면 UI Thread에 화면 표시를 부탁하던지 또는 Handler와 같이 메시지를 이용해 일을 시키는 방법이 있음



The screenshot displays the Android Studio interface. The top pane shows a Java code snippet within a `while(true)` loop, where a `TextView` is being updated with a new value. The bottom pane shows the Logcat window with a `FATAL EXCEPTION: Thread-4` error. The error message indicates that the thread is attempting to modify the UI from a non-UI thread, which is not allowed in Android.

```
26 while(true) {
33     number++;
34     Log.i( tag: "test", msg: "thread running..." + number);
35
36     textView.setText(String.valueOf(number));
37
38 }
39
40
41 };
```

Logcat

Emulator Nexus\_5X\_API\_27\_0 com.iot.threadtest1 (416) Error

05-02 01:44:48.716 416-434/com.iot.threadtest1 E/AndroidRuntime: FATAL EXCEPTION: Thread-4  
Process: com.iot.threadtest1, PID: 416  
android.view.ViewRootImpl\$CalledFromWrongThreadException: Only the original thread that created a  
at android.view.ViewRootImpl.checkThread(ViewRootImpl.java:7313)  
at android.view.ViewRootImpl.requestLayout(ViewRootImpl.java:1161)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.view.View.requestLayout(View.java:21995)  
at android.widget.TextView.checkForRelayout(TextView.java:8531)  
at android.widget.TextView.setText(TextView.java:5394)  
at android.widget.TextView.setText(TextView.java:5250)  
at android.widget.TextView.setText(TextView.java:5207)  
at com.iot.threadtest1.MainActivity\$1.run(MainActivity.java:36)

Terminal Build Logcat Android Profiler Run TODO

Gradle build finished in 1s 595ms (2 minutes ago) 38:18 CRLF UTF-8 Context: <no context>



- 화면에 접근할 때 비동기적으로 접근하는 부분에 유의

```
package com.iot.threadtest1;

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.widget.TextView;

public class MainActivity extends AppCompatActivity { // 메인 화면
    private Thread thread;
    private TextView textView;
    private int number = 0;

    @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
        textView = (TextView)findViewById(R.id.textview);
    }
}
```

```
thread = new Thread() {
    @Override // 부모 메소드 재정의
    public void run() {
        super.run();
        while(true) {
            try {
                sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            number++;
            Log.i("test", "thread running..." + number);
            runOnUiThread(new Runnable() {
                @Override // 부모 메소드 재정의
                public void run() {
                    textView.setText(String.valueOf(number));
                }
            });
        }
    }
};
thread.start();
}
```

- 쓰레드를 1개 더 추가하고, 1초에 1씩 감소하도록 구현하고 테스트 해보자. 또한 메시지를 핸들러에게 전달해서 핸들러가 화면을 갱신하도록 구현