

안드로이드 SDK 개발 플랫폼 활용하기

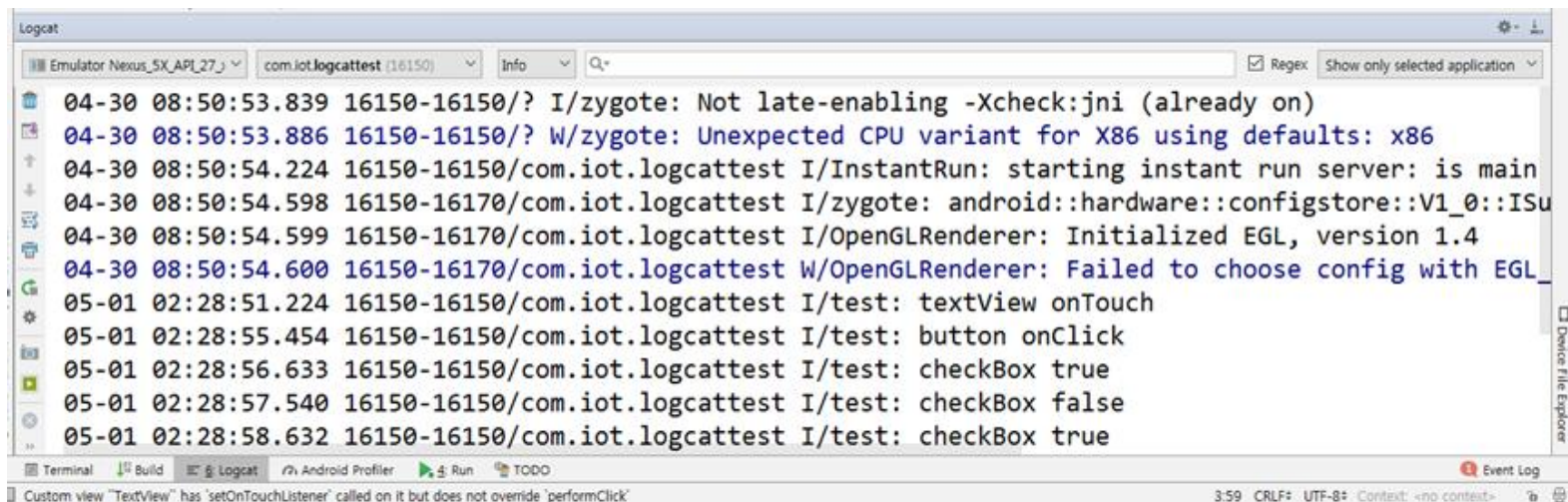
로그켓을 활용한 디버깅

- Log (로그)
 - 사전적인 의미로는 장작, 쌓아둔다는 의미
 - 향해 일지와 같이 실행 내역을 모두 기록하는 목적으로 사용
- Log 클래스의 static 메소드
 - e : 높음
 - v : 낮음

종 류	의 미
Log.e(String tag, String msg)	Error 로그
Log.w(String tag, String msg)	Warning 로그
Log.i(String tag, String msg)	Information 로그
Log.d(String tag, String msg)	Debug 로그
Log.v(String tag, String msg)	Verbose 로그

로그의 수준과 의미

- 논리적인 오류는 해결이 어려움
 - LogCat과 Debugger를 이용하면 쉽게 해결 가능
- LogCat
 - 앱이 동작되는 중에 로그를 실시간으로 확인할 수 있는 도구
 - 앱이 실행시점에 비정상적으로 종료 시 원인 규명에 사용
 - 로그캣의 예외 사항을 확인
 - 몇 번째 줄에서 비정상 종료되었는지 확인 가능
 - 문제의 원인도 알아 낼 수 있음



```
04-30 08:50:53.839 16150-16150/? I/zygote: Not late-enabling -Xcheck:jni (already on)
04-30 08:50:53.886 16150-16150/? W/zygote: Unexpected CPU variant for X86 using defaults: x86
04-30 08:50:54.224 16150-16150/com.iot.logcattest I/InstantRun: starting instant run server: is main
04-30 08:50:54.598 16150-16170/com.iot.logcattest I/zygote: android::hardware::configstore::V1_0::ISu
04-30 08:50:54.599 16150-16170/com.iot.logcattest I/OpenGLRenderer: Initialized EGL, version 1.4
04-30 08:50:54.600 16150-16170/com.iot.logcattest W/OpenGLRenderer: Failed to choose config with EGL
05-01 02:28:51.224 16150-16150/com.iot.logcattest I/test: textView onTouch
05-01 02:28:55.454 16150-16150/com.iot.logcattest I/test: button onClick
05-01 02:28:56.633 16150-16150/com.iot.logcattest I/test: checkBox true
05-01 02:28:57.540 16150-16150/com.iot.logcattest I/test: checkBox false
05-01 02:28:58.632 16150-16150/com.iot.logcattest I/test: checkBox true
```

Custom view "TextView" has "setOnTouchListener" called on it but does not override "performClick"

3:59 CRLF: UTF-8: Context: <no context>

- 로그
 - 안드로이드 앱의 시작부터 끝까지 행적이 남는 곳
 - 예전에 발생했던 문제의 원인을 찾기도 함
 - 디버거와는 목적과 시점에 서 차이가 있음
 - 디버거: 예상되는 문제 지역에 break point 설정 후 확인 (어려운 문제 시)
 - 로그캣: 앱이 이미 죽고 난 이후 원인을 찾을 때 사용 (편해서 많이 씀)
- 로그 표시 메소드들은 모두 static 메소드
 - Log 클래스의 인스턴스를 생성할 필요가 없음
 - 단순히 주어진 문자열을 로그캣으로 표시하는 역할
 - 어디서든 인스턴스를 생성하지 않고 쉽게 표시하기 위한 목적

- 텍스트뷰와 버튼, 체크박스를 가지는화면을 만들고 각 이벤트가 발생 할 때 마다 상태를 로그에 출력
 - i 등급의 로그 메시지를 남기도록 하며, tag는 통일해서 기입

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="안드로이드 APK 파일의
스키마 경로는 생략합니다."
    xmlns:app="안드로이드 APK 파일의 스키마 경로는 생략"
    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" // 컨텐츠에 맞게 최소화
        android:layout_height="wrap_content" // 컨텐츠에 맞게 최소화
        android:text="Hello World!"
        android:textSize="28dp"
        android:textColor="#000000"
        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" // 콘텐츠에 맞게 최소화
    android:layout_height="wrap_content" // 콘텐츠에 맞게 최소화
    android:text="Push"
    android:textSize="28dp"
    android:textColor="#000000"
    app:layout_constraintTop_toBottomOf="@id/textView"
    app:layout_constraintLeft_toLeftOf="@id/textView"
    app:layout_constraintRight_toRightOf="@id/textView"
/>
<CheckBox
    android:id="@+id/checkbox"
    android:layout_width="wrap_content" // 콘텐츠에 맞게 최소화
    android:layout_height="wrap_content" // 콘텐츠에 맞게 최소화
    app:layout_constraintTop_toBottomOf="@id/button"
    app:layout_constraintLeft_toLeftOf="@id/button"
    app:layout_constraintRight_toRightOf="@id/button"
/>
</android.support.constraint.ConstraintLayout>
```

- TextView와 Button외에 CheckBox 뷰를 추가
- CheckBox의 리스너는 `onCheckedChangeListener`가 제공
 - 파라미터로 상태 변화를 알 수 있음
 - 어떤 선택을 했는지도 `boolean` 타입으로 알 수 있음
 - 최근 설정 정보를 기억하고 싶다면 파일이나 Preference, 또는 DBMS를 이용할 수 있음



- MainActivity.java
 - 가장 최근의 로그일 수록 아랫 쪽에 표시되기 때문에 먼저 기존 로그를 모두 삭제하고, 프로그램을 실행

```
package com.iot.logcattest;
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.MotionEvent;
import android.view.View; // 뷰계열 최상위 클래스
import android.widget.Button; // 버튼 뷰 사용
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
import org.w3c.dom.Text;
public class MainActivity extends AppCompatActivity { // 메인 화면
    private TextView textView;
    private Button button;
    private CheckBox checkBox;
```



```

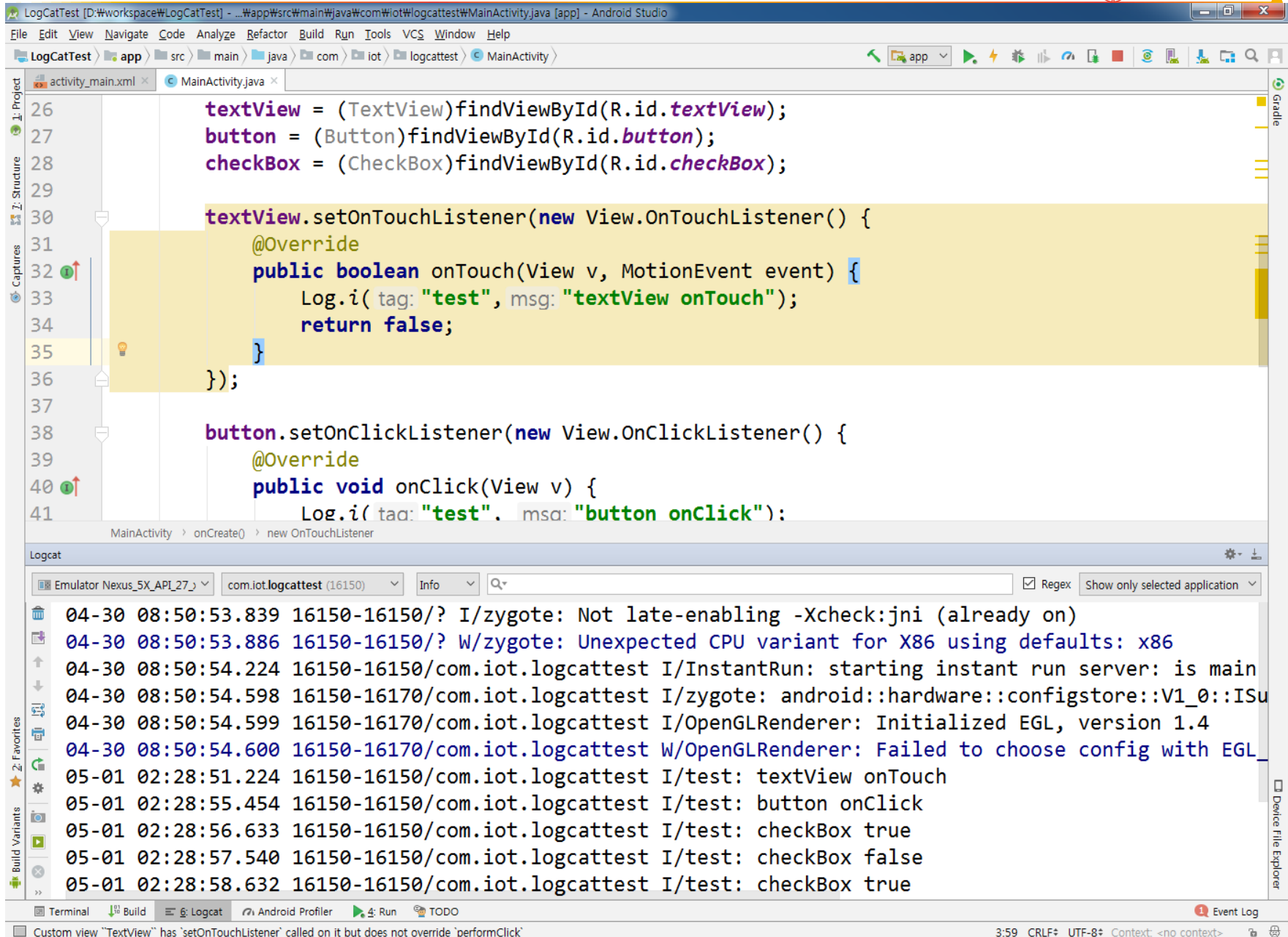
@Override // 부모 메소드 재정의
protected void onCreate(Bundle savedInstanceState) { // 화면생성 이벤트
    super.onCreate(savedInstanceState); // 부모 생성자 호출
    setContentView(R.layout.activity_main); // 메인 화면 표시
    textView = (TextView)findViewById(R.id.textView);
    button = (Button)findViewById(R.id.button);
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

    textView.setOnTouchListener(new View.OnTouchListener() {
        @Override // 부모 메소드 재정의
        public boolean onTouch(View v, MotionEvent event) {
            Log.i("test", "textView onTouch");
            return false;
        }
    });

    button.setOnClickListener(new View.OnClickListener() { // 클릭리스터 생성
        @Override // 부모 메소드 재정의
        public void onClick(View v) { // 클릭 이벤트 처리
            Log.i("test", "button onClick");
        }
    });
    
```

```
checkBox.setOnCheckedChangeListener(new  
CompoundButton.OnCheckedChangeListener() {  
  
    @Override // 부모 메소드 재정의  
    public void onCheckedChanged(CompoundButton buttonView, boolean  
                                isChecked) {  
  
        Log.i("test","checkBox "+isChecked);  
    }  
  
});  
}  
}
```

로그캣 실행 화면



The screenshot displays the Android Studio interface. The top toolbar includes icons for Run, Build, and other development tools. The main editor shows the `MainActivity.java` file with the following code:

```
26 textView = (TextView)findViewById(R.id.textView);
27 button = (Button)findViewById(R.id.button);
28 checkBox = (CheckBox)findViewById(R.id.checkBox);
29
30 textView.setOnTouchListener(new View.OnTouchListener() {
31     @Override
32     public boolean onTouch(View v, MotionEvent event) {
33         Log.i( tag: "test", msg: "textView onTouch");
34         return false;
35     }
36 });
37
38 button.setOnClickListener(new View.OnClickListener() {
39     @Override
40     public void onClick(View v) {
41         Log.i( tag: "test", msg: "button onClick");
42     }
43 });
```

The Logcat window at the bottom shows the following log output:

```
04-30 08:50:53.839 16150-16150/? I/zygote: Not late-enabling -Xcheck:jni (already on)
04-30 08:50:53.886 16150-16150/? W/zygote: Unexpected CPU variant for X86 using defaults: x86
04-30 08:50:54.224 16150-16150/com.iot.logcattest I/InstantRun: starting instant run server: is main
04-30 08:50:54.598 16150-16170/com.iot.logcattest I/zygote: android::hardware::configstore::V1_0::ISU
04-30 08:50:54.599 16150-16170/com.iot.logcattest I/OpenGLRenderer: Initialized EGL, version 1.4
04-30 08:50:54.600 16150-16170/com.iot.logcattest W/OpenGLRenderer: Failed to choose config with EGL_
05-01 02:28:51.224 16150-16150/com.iot.logcattest I/test: textView onTouch
05-01 02:28:55.454 16150-16150/com.iot.logcattest I/test: button onClick
05-01 02:28:56.633 16150-16150/com.iot.logcattest I/test: checkBox true
05-01 02:28:57.540 16150-16150/com.iot.logcattest I/test: checkBox false
05-01 02:28:58.632 16150-16150/com.iot.logcattest I/test: checkBox true
```

The bottom status bar shows the time as 3:59, CRLF+ UTF-8+ Context: <no context>.

- 로그캣 필터 사용해 보자.
 - 안드로이드 스튜디오의 로그캣에서는 수준별 필터링, 태그별 필터링, 앱별 필터링 기능을 제공
 - 보고 싶은 로그만 필터링 해서 표시
- 로그를 필터링 할때 프로세스 아이디를 이용해서 필터링 가능
 - 기존 로그 메시지를 모두 삭제하고 싶다면 Clean Message를 클릭
- 동작 중에 에러가 발생하는 경우 붉은 색 글씨와 함께 예외가 발생한 곳의 위치와 backtrace 기능도 제공
 - 프로그램의 흐름을 이해하고 분석하는 목적으로 사용