Bab 5 | Background Thread

Background thread adalah jalur kerja tambahan dalam aplikasi yang berjalan di luar jalur utama (main thread), untuk mengerjakan tugas-tugas berat tanpa mengganggu responsivitas aplikasi.

Di Android, background thread digunakan untuk:

- Menghindari aplikasi nge-lag atau freeze saat melakukan operasi berat seperti:
    - Mengunduh data dari internet
    - Membaca atau menulis file besar
    - Akses ke database
    - Pemrosesan gambar atau video
- Karena Android mengharuskan semua interaksi dengan tampilan (UI) dilakukan di main thread, dan semua tugas berat harus dipindahkan ke background thread.

5.1 Thread

Thread adalah unit terkecil dari eksekusi program.
Di dalam satu aplikasi, kamu bisa menjalankan beberapa thread sekaligus untuk mengolah tugas-tugas berbeda secara paralel atau bersamaan.

Gampangnya, thread itu seperti jalur kerja.
Kalau programmu adalah sebuah pabrik, maka thread adalah pekerja yang menyelesaikan tugas-tugas tertentu.

Contoh analogi sederhana:

- Satu aplikasi (proses) = satu pabrik.
- Banyak thread = banyak pekerja di pabrik, masing-masing mengerjakan bagian tugas tertentu.

Dalam Android

- Main Thread: thread utama, tugasnya khusus untuk mengurus tampilan (UI).

- Background Thread: thread tambahan, tugasnya menjalankan pekerjaan berat (seperti download data, baca file besar) supaya main thread tidak terganggu.

Android mewajibkan semua perubahan UI hanya boleh dilakukan dari Main Thread.

Contoh pengunaan thread :

Membuat Thread Manual

Main.java

```
new Thread(new Runnable() {
    @Override
    public void run() {
        // Ini dijalankan di background thread
        for (int i = 0; i < 5; i++) {
            System.out.println("Sedang bekerja di background: " + i);

            try {
                Thread.sleep(1000); // Istirahat 1 detik
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }

        // Kalau mau update tampilan, harus kembali ke Main Thread
    }
}).start();
```

Penjelasan:

- `new Thread(...).start();` artinya membuat thread baru dan langsung memulai jalan.
- `run()` adalah apa yang mau dikerjakan di thread itu.
- `Thread.sleep(1000);` untuk pause 1 detik (1000 milidetik) tiap putaran loop.

Mengupdate UI dari Background Thread

Penjelasan:

- Operasi berat (2+2, contoh sederhana) dikerjakan di thread latar belakang.
- Setelah selesai, pakai `runOnUiThread` untuk update tampilan di Main Thread.
- Penting: Kalau tidak kembali ke Main Thread, aplikasi bisa crash saat ubah UI.

6.2 Handler

Handler adalah alat bantu di Android untuk:

- Mengirim dan menerima pesan (message) atau tugas (runnable) antara thread,
- Mengelola antrian tugas dalam sebuah thread, biasanya Main Thread atau Background Thread

Dengan Handler, kita bisa:

- Menunda eksekusi kode.
- Menjalankan kode dari background thread ke main thread (misal: update tampilan).
- Membuat sistem antrian tugas (task queue) sederhana.

Contoh Pengunaan Handler :

Menjalankan sesuatu dengan delay 3 detik

Main.java

```java
new Handler(Looper.getMainLooper()).postDelayed(new Runnable() {
    @Override
    public void run() {
        // Ini dijalankan di Main Thread setelah 3 detik
        TextView textView = findViewById(R.id.textView);
        textView.setText("3 Detik telah berlalu!");
    }
}, 3000); // 3000 ms = 3 detik
```

Penjelasan:

- `postDelayed`: menjalankan sesuatu nanti setelah sekian waktu.
- `Looper.getMainLooper()`: artinya Handler ini terhubung ke Main Thread.

Mengirim tugas dari Background ke Main Thread

Main.java

```java
new Thread(new Runnable() {
    @Override
    public void run() {
        // Kerjaan berat di background
        final String hasil = "Data sudah diambil!";

        // Kirim ke Main Thread
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            @Override
            public void run() {
                TextView textView = findViewById(R.id.textView);
                textView.setText(hasil);
            }
        });
    }
}).start();
```

Penjelasan:

- `post()`: langsung menjalankan kode di Main Thread, secepat mungkin.
- Cocok untuk update UI setelah background task.

6.3 Executor

Executor adalah komponen Java (dan Android) yang berfungsi untuk mengelola eksekusi tugas-tugas (Runnable) di thread secara lebih terstruktur dan efisien.

Kalau pakai `Thread` manual, kamu harus:

- Bikin thread satu-satu.
- Jalankan satu-satu.
- Kelola sendiri kapan selesai, kapan mati.

Kalau pakai Executor, semua itu dikelola otomatis:

- Bisa membuat banyak thread.
- Bisa atur berapa banyak tugas jalan bersamaan.
- Bisa antri tugas yang belum sempat jalan.

Contoh Pengunaan Executor :

Main.java

```
// Import dulu
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

// Buat Executor
ExecutorService executorService = Executors.newSingleThreadExecutor();

// Kirim tugas ke Executor
executorService.execute(new Runnable() {
    @Override
    public void run() {
        // Ini berjalan di thread lain (bukan main thread)
        int hasil = 5 + 5;

        // Kalau mau update UI, harus kembali ke Main Thread
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                TextView textView = findViewById(R.id.textView);
                textView.setText("Hasil: " + hasil);
            }
        });
    }
});

// Jangan lupa tutup kalau sudah tidak dipakai
executorService.shutdown();
```

Tipe - tipe pada executor :

Tipe	Penjelasan
newSingleThreadExecutor()	Selalu pakai satu thread.
newFixedThreadPool(int n)	Pool dengan jumlah thread tetap (misal 5 thread).
newCachedThreadPool()	Dinamis dengan bisa menggunakan thread baru atau hapus thread idle.

6.4 Penerapan Background Thread pada aplikasi android :

striings.xml

```xml
<resources>
    <string name="app_name">File Download</string>
    <string name="start_task">Start Download</string>
    <string name="status">Downloading...</string>
    <string name="download_completed">Download Completed!</string>
    <string name="downloading">Downloading %d %%</string>
</resources>
```

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="vertical"
        android:padding="16dp">

        <Button
            android:id="@+id/btn_start"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="300dp"
            android:text="@string/start_task" />

        <TextView
            android:id="@+id/tv_status"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            tools:text="@string/status" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="400dp"
            android:text="@string/hidden_text" />

    </LinearLayout>

</ScrollView>
```

MainActivity.java

```java
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnStart = findViewById(R.id.btn_start);
        TextView tvStatus = findViewById(R.id.tv_status);

        // Executor untuk menjalankan background task
        ExecutorService executor = Executors.newSingleThreadExecutor();
        Handler handler = new Handler(Looper.getMainLooper());

        btnStart.setOnClickListener(v -> {
            executor.execute(() -> {
                try {
                    // Simulasi proses download di background thread
                    for (int i = 0; i <= 10; i++) {
                        Thread.sleep(500); // Delay simulasi pengunduhan
                        int percentage = i * 10;

                        handler.post(() -> {
                            // Update UI di main thread
                            if (percentage == 100) {
                                tvStatus.setText(R.string
.download_completed);
                            } else {
                                tvStatus.setText(
                                    String.format(getString(R.string
.downloading), percentage)
                                );
                            }
                        });
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            });
        });
    }
}
```