

Tutorial

Layout Aplikasi Android

oleh: [Mohamad Sani \(sani@msani.net\)](mailto:sani@msani.net)
25 Oktober 2010

NB: Tutorial ini adalah lanjutan dari tutorial sebelumnya.

Layout adalah rancangan antarmuka (*user interface*) pada sebuah Activity. Pada program HelloWorld sebelumnya, kita membuat layout antarmuka aplikasi melalui kode program. Pendekatan ini tidak dianjurkan, apalagi jika aplikasi yang dibuat tergolong besar, karena perubahan layout sedikit saja sangat rumit diterapkan melalui kode.

Oleh karena itu, Android menyediakan alternatif membuat layout dengan XML. Berkas layout XML secara otomatis terhubung dengan kelas view. Kelebihan membuat layout dengan XML adalah kita dapat memisahkan antara tampilan (kita urus melalui XML) dan program (kita urus melalui kode). Dengan demikian, jika kita ingin mengubah tampilan, kita tidak perlu mengganggu kode. Kegunaannya lebih terasa ketika kita ingin membuat antarmuka yang berbeda untuk orientasi layar berbeda atau untuk *smartphone* berbeda.

1 Membuat Layout XML

1. Pada Eclipse Package Explorer, buka `/res/layout/main.xml`
2. Ganti isi `main.xml` dengan XML berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"/>
```

3. Save `main.xml`.

Struktur layout XML Android sangat sederhana: berupa tree dari elemen-elemen XML dimana setiap node-nya adalah subkelas dari kelas `View`. Pada contoh di atas `main.xml` hanya terdiri dari satu elemen `View` yaitu `TextView`. Kita bisa menggunakan kelas apapun yang meng-*extend* `View` sebagai elemen pada layout XML termasuk kelas yang kita buat sendiri jika kita definisikan meng-*extend* `View`. Struktur sederhana seperti ini menjadikan pembuatan antarmuka cepat dan gampang.

Pada contoh `main.xml` di atas, `TextView` memiliki 5 atribut XML. Berikut adalah penjelasan singkatnya:

- **xmlns:android**. Deklarasi *namespace* XML untuk memberitahu *tools* Android bahwa kita akan merujuk ke atribut-atribut yang didefinisikan dalam *namespace* Android. Tag paling luar dari setiap layout Android harus memiliki atribut ini.
- **android:id**. Atribut ini memberikan *identifier* unik ke elemen `TextView`. Kita dapat menggunakan id tersebut untuk memanggil layout `View` ini dari kode atau dari *resource*

XML lain.

- **android:layout_width**. Atribut ini mendefinisikan seberapa banyak lebar layar yang akan digunakan untuk `View` ini. Kita mengisinya dengan “`fill_parent`” yang artinya kita akan menggunakan seluruh lebar layar.
- **android:layout_height**. Sama seperti `android:layout_width` tapi untuk variabel tinggi.
- **android:text**. Mendefinisikan teks yang akan ditampilkan oleh `TextView`. Pada contoh di atas, kita menggunakan suatu *resource* string, bukan mendeklarasikan string secara langsung. String `hello` didefinisikan pada `res/values/strings.xml`. Cara inilah yang direkomendasikan karena akan membuat lokalisasi* mudah. Kita tidak harus mengubah layout bila ingin merubah suatu *resource*.

Seluruh layout XML anda taruh dalam direktori `res/layout`. “`res`” adalah singkatan dari “`resource`”. Di sinilah anda menaruh seluruh aset-aset yang tidak berupa kode, bisa berupa gambar, musik, string dsb.

Plugin Eclipse membuat `main.xml` secara otomatis. Dalam tutorial sebelumnya, meskipun ada kita tidak menggunakannya karena tutorial tersebut bermaksud menjelaskan tentang *framework* Android. Ke depannya, kita seharusnya selalu mendefinisikan layout pada sebuah XML bukan pada kode.

2 Membuat Resource String

1. Buka `/res/values/strings.xml`

2. Ubah string `hello` menjadi “`Hello, world! Ini resource string!`” sehingga eseluruhan isi `strings.xml` menjadi:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello, world! I am a string resource!</string>
  <string name="app_name">Hello, World</string>
</resources>
```

3. save `strings.xml`.

3 Mengganti Pembuatan Layout Menjadi Menggunakan XML

Buka kelas `HelloWorld` dan ubah sehingga keseluruhan `HelloWorld.java` akan menjadi:

```
package com.example.helloworld;

import android.app.Activity;
import android.os.Bundle;

public class HelloWorld extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Ketika mengubah, cobalah fitur code-completion Eclipse. Ketika sedang mengetik “`R.layout.main`” plugin Eclipse akan menawarkan usulan. Fitur ini akan sangat berguna di banyak situasi.

Pada tutorial lalu kita memberikan `setContentView()` sebuah objek `View`. Sekarang kita

merujuknya ke *resource* layout yaitu `R.layout.main` yang merupakan kompilasi objek yang kita buat di `/res/layout/main.xml`. Plugin Eclipse secara otomatis membuat referensi tersebut di kelas `R.java` yang berada di direktori `/gen`.

Karena sebelumnya kita sudah membuat konfigurasi, tinggal klik **Run > Run History > Android Activity** untuk menjalankan aplikasi. Selain perubahan pada string `TextView`, tidak ada perbedaan lain pada aplikasi. Namun, kali ini kita telah menggunakan pendekatan layout yang berbeda.

4 Membuat Activity untuk View Google Map

1. Buat AVD baru dengan build target "Google APIs - API Level 8"
2. Jalankan Virtual Device tersebut.
3. Buat proyek baru bernama **LatihanGoogleMaps**
4. Klik kanan **AndroidManifest.xml**, pilih **Open with > Android Layout Editor**
5. Tambahkan kode berikut di dalam elemen `<application>`

```
<uses-library android:name="com.google.android.maps" />
```

6. Tambahkan kode berikut di dalam elemen `<manifest>`

```
<uses-permission android:name="android.permission.INTERNET" />
```

7. Tambahkan kode berikut di dalam elemen `<activity>`

```
android:theme="@android:style/Theme.NoTitleBar"
```

8. Buka **main.xml**, ganti isinya dengan kode berikut

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mapview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="Isi API Key peta anda di sini"
/>
```

9. Buka **LatihanGoogleMaps.java**, ganti `extend Activity` menjadi `extend MapActivity`
10. Pastikan anda telah mengimpor `MapActivity`, nantinya anda juga butuh mengimpor `MapView`

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
```

11. Tambahkan method berikut di dalam kelas **LatihanGoogleMaps**

```
@Override
protected boolean isRouteDisplayed() {
    return false;
}
```

Android menyediakan banyak *template* Layout dan View yang bisa tinggal pakai. Pada tutorial sebelumnya kita telah mencoba layout Linier Layout. Di sini, kita akan mencoba salah satu View yaitu Google Map View karena View jenis ini agak unik dibanding View yang lain.

[Langkah 1, 2 & 3] Kita harus membuat Android Virtual Device (AVD) yang sama dengan target *deployment* aplikasi, dalam tutorial ini kita menggunakan Google API Level 8. AVD butuh waktu lama untuk booting, ada baiknya kita menjalankannya terlebih dahulu. Sambil menunggu AVD

booting, kita lanjut membuat program.

[Langkah 5] Karena pustaka (*library*) **Maps** bukan merupakan pustaka standar Android, kita harus mendeklarasikannya di **AndroidManifest.xml**.

[Langkah 6] Nantinya kita butuh akses internet untuk mengambil data-data peta. Langkah ini berarti kita meminta izin (*permission*) akses internet.

[Langkah 7] Anggapannya (*by default*), aplikasi akan menampilkan nama aplikasi pada layar, karena kita hendak membuat **View** berupa peta, kita menyingkirkan **TitleBar** ini agar layar lebih lebar.

[Langkah 8] atribut **android:clickable** berarti kita mengizinkan pengguna berinteraksi dengan peta. Bila diisi **false**, ketika peta disentuh tidak akan terjadi apa-apa.

Atribut **android:apiKey** diisi dengan API Key aplikasi kita. API Key merupakan sertifikat bahwa aplikasi kita terdaftar untuk menggunakan Maps Service. API Key dibutuhkan untuk mendapatkan data peta, meskipun kita masih sedang mengembangkan aplikasinya. Pendaftarannya gratis dan hanya sebentar. Untuk informasi lebih lanjut, lihat <http://code.google.com/android/add-ons/google-apis/mapkey.html>

NB: Untuk saat ini, isi saja nilainya dengan **api.samples**.

[Langkah 9] **MapActivity** adalah subkelas khusus dari **Activity** yang menyediakan pustaka perpetaan.

[Langkah 11] Di dalam setiap **MapActivity**, method **isRouteDisplayed()** harus ada. Method ini dibutuhkan agar Maps Service dapat mengkalkulasi informasi rute. Saat ini kita tidak membutuhkannya, oleh karena itu kita isi nilainya dengan **false**.