ANDROID DEVELOPMENT

By Muhammad Aufa Rijal / 19040079 / 4D

1. Manajemen memori pada Aplikasi Android

Android Runtime (ART) dan mesin virtual Dalvik menggunakan paging dan memory mapping (mmapping/pemetaan memori) untuk mengelola memori. Ini berarti setiap memori yang dimodifikasi oleh aplikasi, baik dengan mengalokasikan objek baru atau mengakses halaman yang di-mmap, tetap berada di dalam RAM dan tidak dapat di-page out. Satu-satunya cara untuk membebaskan memori dari aplikasi adalah dengan melepaskan referensi objek yang digunakan aplikasi, sehingga memori tersedia bagi pembersih sampah memori. Namun itu dengan satu pengecualian: setiap file yang di-mmap masuk tanpa modifikasi, misalnya kode, dapat di-page out dari RAM jika sistem ingin menggunakan memori tersebut di tempat lain.

a. Pembersihan sampah memori

Lingkungan memori terkelola, seperti ART atau mesin virtual Dalvik, melacak setiap alokasi memori. Setelah menentukan bahwa memori tidak lagi digunakan oleh program, lingkungan memori terkelola akan mengembalikan memori tersebut ke heap, tanpa intervensi apa pun dari programmer. Mekanisme untuk mengklaim kembali memori yang tidak terpakai dalam lingkungan memori terkelola ini disebut *pembersihan sampah memori*. Pembersihan sampah memori memiliki dua tujuan: menemukan objek data dalam program yang tidak dapat diakses di waktu mendatang, dan mengklaim kembali resource yang digunakan oleh objek tersebut.
Heap memori Android bersifat generasional, yang berarti ada berbagai bucket alokasi yang dilacak berdasarkan ekspektasi umur dan ukuran objek yang dialokasikan. Misalnya, objek yang baru dialokasikan termasuk dalam *Generasi muda*. Jika sebuah objek tetap aktif cukup lama, objek tersebut dapat dipromosikan ke generasi yang lebih tua, yang diikuti oleh generasi permanen.
Setiap generasi heap memiliki batas atas khusus terkait banyaknya memori yang dapat ditempati oleh objek yang ada di sana. Setiap kali generasi mulai terisi, sistem akan menjalankan peristiwa pembersihan sampah memori sebagai upaya untuk mengosongkan memori. Durasi pembersihan sampah memori bergantung pada generasi objek yang dibersihkan dan banyaknya objek aktif di setiap generasi.
Meskipun pembersihan sampah memori berlangsung sangat cepat, proses ini tetap dapat memengaruhi performa aplikasi Anda. Secara umum, Anda tidak dapat mengontrol kapan peristiwa pembersihan sampah memori terjadi dari dalam kode. Sistem memiliki sekumpulan kriteria aktif untuk menentukan kapan pembersihan sampah memori harus dijalankan. Saat kriteria ini terpenuhi, sistem akan berhenti menjalankan proses dan memulai pembersihan sampah memori. Jika pembersihan sampah memori terjadi di tengah-tengah loop pemrosesan yang intensif, seperti animasi atau selama pemutaran musik, pembersihan sampah memori dapat meningkatkan waktu proses. Peningkatan ini berpotensi mendorong eksekusi kode

di aplikasi Anda melampaui ambang batas 16 milidetik yang disarankan untuk proses rendering frame yang efisien dan lancar.

Selain itu, alur kode Anda dapat menjalankan jenis pekerjaan yang memaksa terjadinya pembersihan sampah memori lebih sering, atau menjadikannya berlangsung lebih lama daripada biasanya. Misalnya, jika Anda mengalokasikan beberapa objek di bagian terdalam for-loop selama setiap frame animasi pencampuran alfa, Anda dapat mengotori heap memori dengan banyak objek. Dalam situasi tersebut, pembersih sampah memori akan menjalankan beberapa peristiwa pembersihan sampah memori, dan hal itu dapat menurunkan performa aplikasi Anda.

b. Membagikan memori (Shared Memory)

Agar sesuai dengan semua yang dibutuhkan di RAM, Android mencoba membagikan halaman RAM ke berbagai proses. Android melakukannya dengan cara berikut:

- Setiap proses aplikasi diambil dari proses yang sudah ada, yang disebut Zygote. Proses Zygote dimulai ketika sistem melakukan booting dan memuat kode framework dan resource yang umum (seperti tema aktivitas). Untuk memulai proses aplikasi baru, sistem akan mengambil proses Zygote, lalu memuat dan menjalankan kode aplikasi dalam proses baru. Dengan pendekatan ini, sebagian besar halaman RAM yang dialokasikan untuk kode framework dan resource dapat dibagikan ke semua proses aplikasi.
- Sebagian besar data statis di-mmap masuk ke dalam proses. Dengan teknik ini, data dapat dibagikan ke berbagai proses, juga memungkinkan data untuk di-page out saat diperlukan. Contoh data statis meliputi: kode Dalvik (dengan menempatkannya dalam file .odex yang ditautkan sebelumnya untuk mmapping langsung), resource aplikasi (dengan mendesain tabel resource menjadi struktur yang dapat di-mmap, dan dengan menyelaraskan entri zip APK), serta elemen project tradisional seperti kode native dalam file .so.
- Di banyak tempat, Android berbagi RAM dinamis yang sama dengan berbagai proses menggunakan region memori bersama yang dialokasikan secara eksplisit (dengan ashmem atau gralloc). Misalnya, tampilan jendela menggunakan memori bersama antara aplikasi dan compositor layar, dan buffer kursor menggunakan memori bersama antara penyedia konten dan klien.

Karena penggunaan memori bersama yang ekstensif, diperlukan kecermatan untuk menentukan banyaknya memori yang digunakan aplikasi. Teknik untuk menentukan penggunaan memori aplikasi dengan tepat dibahas dalam Menyelidiki Penggunaan RAM.

c. Mengalokasikan dan mengklaim kembali memori aplikasi

Heap Dalvik dibatasi ke satu rentang memori virtual untuk setiap proses aplikasi. Hal ini menentukan ukuran heap logika, yang dapat meningkat sesuai kebutuhan, tetapi hanya sebatas yang ditetapkan sistem untuk setiap aplikasi.
Ukuran heap logis tidak sama dengan jumlah memori fisik yang digunakan oleh heap. Saat memeriksa heap aplikasi, Android menghitung sebuah nilai yang disebut Proportional Set Size (PSS, Ukuran Set Proporsional), yang memperhitungkan halaman kotor dan bersih yang digunakan bersama dengan proses lain tetapi hanya sejumlah yang proporsional dengan banyaknya aplikasi yang menggunakan bersama RAM tersebut. Total (PSS) ini adalah apa yang dianggap sistem sebagai jejak memori fisik Anda. Untuk informasi lebih lanjut tentang PSS, lihat panduan Menyelidiki Penggunaan RAM.
Heap Dalvik tidak memadatkan ukuran logis heap, yang berarti Android tidak mendefragmentasi heap untuk menutup ruang. Android hanya dapat menyusutkan ukuran logis heap jika ada ruang tidak terpakai di akhir heap. Namun, sistem masih dapat mengurangi memori fisik yang digunakan oleh heap. Setelah pembersihan sampah memori, Dalvik menjalankan heap dan menemukan halaman yang tidak terpakai, lalu mengembalikan halaman tersebut ke kernel menggunakan madvise. Jadi, pasangan alokasi dan dealokasi untuk bagian memori yang besar akan menghasilkan pemerolehan kembali semua (atau hampir semua) memori fisik yang digunakan. Namun, pemerolehan kembali memori dari alokasi kecil dapat menjadi jauh tidak efisien karena halaman yang digunakan untuk alokasi kecil masih dapat dibagikan dengan sesuatu yang belum dibebaskan.

d. Membatasi memori aplikasi

Agar lingkungan multitasking tetap fungsional, Android menetapkan batas pasti ukuran heap untuk setiap aplikasi. Batas ukuran heap pasti ini bervariasi antarperangkat, tergantung banyaknya RAM yang dimiliki perangkat secara keseluruhan. Jika telah mencapai kapasitas heap dan mencoba mengalokasikan lebih banyak memori, aplikasi Anda akan menerima `OutOfMemoryError`.
Dalam beberapa kasus, sebaiknya Anda mengkueri sistem untuk mengetahui secara pasti banyaknya ruang heap yang Anda miliki di perangkat saat ini misalnya, untuk menentukan banyaknya data yang dapat disimpan dengan aman di dalam cache. Untuk mengetahui angkanya, panggil `getMemoryClass()`. Metode ini menampilkan bilangan bulat yang menunjukkan jumlah megabyte yang tersedia untuk heap aplikasi Anda.

e. Beralih aplikasi

Saat pengguna beralih antar-aplikasi, Android akan menyimpan aplikasi yang tidak berjalan di latar depan artinya, yang tidak terlihat oleh pengguna atau menjalankan layanan latar depan, seperti pemutaran musik di dalam cache. Misalnya, saat

pengguna meluncurkan aplikasi untuk pertama kalinya, sebuah proses akan dibuat untuk peristiwa tersebut; tetapi saat pengguna menutup aplikasi, proses tersebut *tidak* berhenti. Sistem menyimpannya di dalam cache. Jika nanti pengguna kembali ke aplikasi itu, sistem akan menggunakan kembali proses tersebut, sehingga peralihan aplikasi berlangsung lebih cepat.

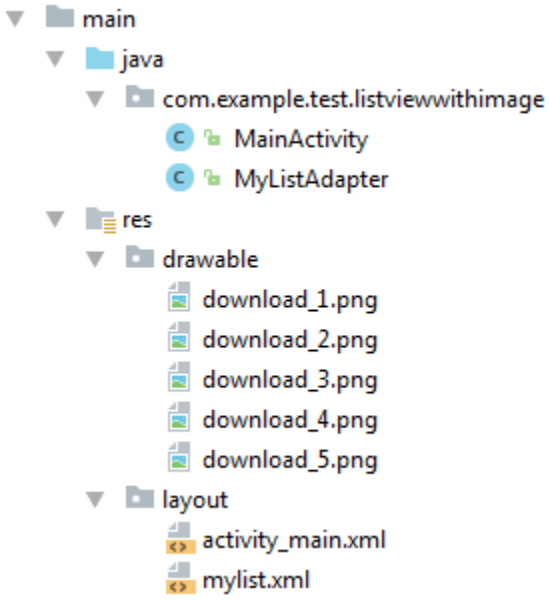
Jika memiliki proses yang tersimpan dalam cache dan mempertahankan resource yang saat ini tidak diperlukan, aplikasi Anda dapat memengaruhi performa sistem secara keseluruhan, bahkan saat pengguna tidak sedang menggunakannya. Jika resource seperti memori hampir penuh, sistem akan mengakhiri proses dalam cache. Sistem juga memperhitungkan proses yang menggunakan memori terbanyak, dan dapat menghentikannya untuk mengosongkan RAM.

2. Custom ListView

Contoh cara membuat custom ListView:

Seperti ListView biasa, custom ListView menggunakan class Adapter yang menambahkan content dari datasource / sumber data (seperti string array,array,database, dll). Adapter menjembatani antara sebuah AdapterViews dan Views lainnya.

Struktur:



activity_main.xml :

Buatlah file activity_main.xml di folder layout.

Konten :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.test.listviewwithimage.MainActivity">

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="50dp">
    </ListView>
</RelativeLayout>
```

Selanjutnya buatlah file tambahan bernama mylist.xml di folder layout yang berisi komponen view yang akan ditampilkan didalam ListView

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <ImageView
        android:id="@+id/icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:padding="5dp" />

    <LinearLayout android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical">
```

.

```
    <TextView
        android:id="@+id/title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Medium Text"
        android:textStyle="bold"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:padding="2dp"
        android:textColor="#4d4d4d" />
    <TextView
        android:id="@+id/subtitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:layout_marginLeft="10dp"/>
    </LinearLayout>
</LinearLayout>
```

Letakkan semua gambar yang dibutuhkan ke folder drawable

Activity Class

File: MainActivity.java

```java
package com.example.test.listviewwithimage;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    ListView list;

    String[] maintitle ={
            "Title 1","Title 2",
            "Title 3","Title 4",
            "Title 5",
    };

    String[] subtitle ={
            "Sub Title 1","Sub Title 2",
            "Sub Title 3","Sub Title 4",
            "Sub Title 5",
    };
```

```
Integer[] imgid={
        R.drawable.download_1,R.drawable.download_2,
        R.drawable.download_3,R.drawable.download_4,
        R.drawable.download_5,
};
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    MyListAdapter adapter=new MyListAdapter(this, maintitle, subtitle,imgid);
    list=(ListView)findViewById(R.id.list);
    list.setAdapter(adapter);


    list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
            // TODO Auto-generated method stub
            if(position == 0) {
                //code specific to first list item
                Toast.makeText(getApplicationContext(),"Place Your First Option Code",Toast.LENGTH_SHORT).show();
            }

            else if(position == 1) {
                //code specific to 2nd list item
                Toast.makeText(getApplicationContext(),"Place Your Second Option Code",Toast.LENGTH_SHORT).show();
            }

            else if(position == 2) {

                Toast.makeText(getApplicationContext(),"Place Your Third Option Code",Toast.LENGTH_SHORT).show();
            }
            else if(position == 3) {

                Toast.makeText(getApplicationContext(),"Place Your Forth Option Code",Toast.LENGTH_SHORT).show();
            }
            else if(position == 4) {

                Toast.makeText(getApplicationContext(),"Place Your Fifth Option Code",Toast.LENGTH_SHORT).show();
            }

        }
    });
  }
}
```

Kustomisasi ListView kita

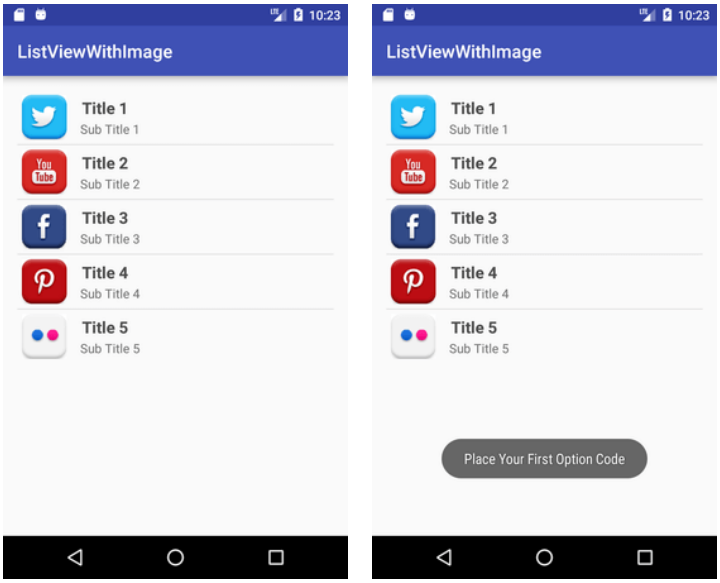
Buat sebuah java class baru dengan nama MyListView.java yang akan extends ke ArrayAdapter class.

MyListView.java

```java
package com.example.test.listviewwithimage;

import android.app.Activity;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MyListAdapter extends ArrayAdapter<String> {

    private final Activity context;
    private final String[] maintitle;
    private final String[] subtitle;
    private final Integer[] imgid;

    public MyListAdapter(Activity context, String[] maintitle,String[] subtitle, Integer[] imgid) {
        super(context, R.layout.mylist, maintitle);
        // TODO Auto-generated constructor stub

        this.context=context;
        this.maintitle=maintitle;
        this.subtitle=subtitle;
        this.imgid=imgid;
    }
```

```
    public View getView(int position,View view,ViewGroup parent) {
        LayoutInflater inflater=context.getLayoutInflater();
        View rowView=inflater.inflate(R.layout.mylist, null,true);

        TextView titleText = (TextView) rowView.findViewById(R.id.title);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);
        TextView subtitleText = (TextView) rowView.findViewById(R.id.subtitle);

        titleText.setText(maintitle[position]);
        imageView.setImageResource(imgid[position]);
        subtitleText.setText(subtitle[position]);

        return rowView;

    };
}
```

Output:



3. **Cara kerja AsyncTask dan AsyncTask Loader**

    a. **AsyncTask**
    Sebuah worker thread adalah thread yang tidak berada pada thread UI atau thread utama. Pakailah class AsyncTask untuk mengimplementasikan sebuah asynchronous, task long-running di worker thread. AsyncTask memberikan kesempatan untuk perform operasi latarbelakang di worker thread dan mempublishkan hasilnya ke thread UI tanpa memerlukan manipulasi threads atau handlers.

    b. **AsnycTask Loader**
    AsyncTaskLoader adalah loader yang sama dengan AsyncTask. AsyncTaskLoader menyediakan sebuah method, yaitu loadInBackground(), yang berjalan pada thread yang terpisah. Hasil dari loadInBackground() secara otomatis dikirimkan ke thread UI, melalui method onLoadFinished() milik callback LoaderManager.

4. **Komponen Broadcast receiver dan notification**

    Android BroadcastReceiver adalah komponen yang terbengkalai dari android yangmana komponen ini mengamati ke system-wide broadcast event atau intents.
    Ketika sesuatu pada events ini terjadi maka aplikasi akan melaksanankan pembuatan notifikasi status bar maupun melakukan sebuah task.
    Tidak seperti activities, Android BroadcastReceiver tidak memiliki user interface. Broadcast receiver umumnya diimplementasikan untuk mendelegasi tasks kepada services bergantung pada tipe intent data yang diterima.

5. **Penggunaan Shared Preferences**

    SharedPreferences adalah salah satu API android yang digunakan untuk menyimpan koleksi kecil dari key-value. Objek SharedPreferences menunjuk pada file yang mengandung key-value dan menyediakan method yang simple untuk membaca dan menulis nya(read & write). Setiap file SharedPreferences di manage oleh framework dan bisa diatur sebagai private atau shared.

6. **Pemanfaatan Android dan Arduino**

    Android juga dapat berinteraksi dengan Arduino, makanya sekarang sudah banyak hasil hasil dari pemanfaatan antara Android dan Arduino. Projek projek yang bisa dibuat menggunakan dengan android dan Arduino :

    - Arduino SmartHome
    - Robot Android Controlled
    - SmartPhone Controlled Mouse
    - Alexa at your fingertips
    - Arduino Push Notification
    - Smart Power Outlet