Modul Perkuliahan

Pemrograman *Mobile*:
Activity dan *Intents*

Dr. Habibullah Akbar

Session 4

2019

SESSION OUTCOMES

1. Mahasiswa melakukan *refreshment* mengenai sesi sebelumnya, yaitu pembahasan 3 tipe layout dan beberapa jenis *View* pada aplikasi Android
2. Mahasiswa dapat menyebutkan dan menjelaskan 4 komponen aplikasi yaitu service, broadcast receiver, content provider, dan aktifitas
3. Mahasiswa dapat memahami konsep aktifitas (*activity*) dan *intent*
4. Mahasiswa dapat membuat aplikasi sederhana yang menerapkan intent untuk melakukan perpindahan antar 2 aktifitas.

OUTLINE MATERI:

1. Pendahuluan
2. Komponent Aplikasi Android
3. Aktifitas (*activity*)
4. *Intent*
5. Latihan Aktifitas dan *Intent*

ISI MATERI

1. Pendahuluan

Pada sesi sebelumnya, anda telah belajar mengenai beberapa jenis layout, khususnya adalah layout *linear* dan layout *relatif.* Selain itu anda juga telah mengenal beberapa jenis Views mencakup *TextView*, *EditText* dan *Button*. Layout ini sebenarnya digunakan oleh komponen aplikasi yang disebut sebagai aktifitas (activity) yaitu tempat dimana *user* berinteraksi dengan aplikasi.

Pada sesi ini, anda akan belajar untuk melakukan perpindahan antara 2 aktifitas, atau 2 layout yang berbeda. Perpindahan dilakukan dengan menggunakan objek yang disebut sebagai *Intent*.

2. Komponen Aplikasi Android

Aplikasi Android biasanya terdiri dari beberapa komponen dasar yang digunakan sebagai pintu masuk bagi *user* ataupun sistem dari luar. Terdapat 4 komponen utama yaitu *services, broadcast receivers, content providers* dan aktifitas.

Komponen pertama adalah *service*. Sebuah *service* adalah sebuah komponen aplikasi yang berjalan di balik layar sehingga *service* tidak memerlukan sebuah tampilan atau layout. *Service* biasanya digunakan untuk operasi yang kompleks ataupun proses yang sudah tidak memerlukan interaksi langsung dengan user. Bahkan *service* dapat terus berjalan walaupun aplikasi yang memicunya telah dimatikan. Contoh penggunaan *service* adalah memainkan suara sedangkan user berinteraksi dengan aplikasi lainnya. Contoh lainnya adalah sinkronisasi dan download data tanpa perlu penanganan langsung dari user.

Komponen kedua adalah *broadcast receiver*. *Broadcast receiver* merupakan komponen aplikasi yang digunakan untuk mengirimkan pesan ataupun siaran tertentu kepada aplikasi lainnya. Menariknya, *broadcast receiver* dapat mengirimkan pesan walaupun aplikasi tujuan tidak sedang berjalan. Contohnya adalah sebuah app yang dapat mengatur alarm untuk memberikan notifikasi kepada user mengenai suatu jadwal tertentu. Namun demikian, *broadcast receiver* juga dapat berasal dari sistem android itu sendiri, misal notifikasi *screen* yang dimatikan, status batere yang mau habis, dan pesan yang telah terkirim. Notifikasi ini biasanya dimunculkan dalam bentuk bar status untuk memberikan info kepada *user* mengenai peristiwa-peristiwa semisal diatas.

Komponen berikutnya adalah *content providers*. Komponen ini berfungsi untuk mengatur data aplikasi yang dapat disimpan kedalam sistem file Android. Data dapat disimpan dalam basis data SQLite, kartu SD, *server web* dan media lainnya. Dengan *content provider*, aplikasi dapat melakukan *query* ataupun modifikasi data, contohnya seperti informasi kontak *user*.

Komponen yang terakhir merupakan fokus dari sesi ini, yaitu aktifitas.

3. Aktifitas (activity)

Aktivitas merupakan tempat dimana user dapat berinteraksi dengan aplikasi. Sebuah aktifitas direpresentasikan dengan sebuah layout, atau *user interface*. Yang perlu diingat adalah sebuah aplikasi seringkali memiliki lebih dari satu aktifitas, misalnya sebuah aplikasi email. Aplikasi email biasanya memiliki aktifitas-aktifitas berikut ini

- 1 aktifitas untuk menampilkan daftar email
- 1 aktifitas untuk membaca sebuah email
- 1 aktifitas untuk membuat email baru
- 1 aktifitas untuk melakukan pengaturan email

Meskipun keempat aktivitas tersebut bekerja sama membentuk pengalaman *user* yang terpadu, namun masing-masing aktifitas sebenarnya tidak bergantung satu sama lain. Dengan demikian, setiap aktifitas tersebut dapat saja berjalan sendirian dan tidak membutuhkan aktifitas lainnya. Bahkan setiap aktifitas tersebut dapat dipanggil oleh aktifitas/aplikasi lainnya. Misalnya, aplikasi kamera dapat memanggil aktivitas membuat email baru (yang sebenarnya merupakan komponen dari aplikasi email) agar user dapat membagikan gambar kepada orang lain.

Setiap kali aktivitas baru dimulai maka aktivitas sebelumnya akan disimpan dalam tumpukan (*backstack*). Ketika aktivitas baru dimulai, aktivitas baru itu akan berada diatas *backstack*, yang mengikuti prinsip "*last in, first out*". Ketika pengguna selesai dengan aktivitas baru tersebut dengan (misalnya) menekan tombol kembali maka aktifitas yang baru ini akan dikeluarkan dari *backstack* dan dihancurkan, kemudian aktivitas sebelumnya akan dilanjutkan kembali.

4. *Intent*

Suatu aktifitas dapat dimulai atau diaktifkan dengan *intent*. *Intent* bisa dilihat sebagai pesan asinkron yang digunakan untuk meminta suatu tindakan dari aktivitas lainnya, ataupun dari komponen aplikasi yang lain. Misalnya, saat user sedang berada pada sebuah aplikasi browser, intent memungkinkan user untuk memanggil suatu aktifitas lainnya seperti melihat map ataupun menjalankan aplikasi kamera.

Biasanya terdapat beberapa kasus penggunaan *intent*:

1. Untuk memulai aktivitas.
Anda bisa memulai aktifitas baru dengan menggunakan *intent*, dengan metode startActivity().

2. Untuk mengirim data antar aktifitas.
Anda dapat mentransfer data antar aktifitas menggunakan *intent*, misal seperti mengirim data gambar ataupun data isian form.

3. Untuk memulai *service*.
Intent juga dapat digunakan untuk memanggil *service*, dan *service* akan berjalan di latar belakang tanpa diperlukan layout/aktifitas.

4. Untuk mengirim *broadcast* (siaran).
Siaran adalah pesan yang bisa diterima sistem Android. Sistem dapat menampilkan berbagai siaran misalnya saat sistem *booting* atau saat perangkat mulai mengisi daya. Anda juga bisa

mengirim siaran ke aplikasi lain dengan meneruskan *intent* ke sendBroadcast(), sendOrderedBroadcast(), atau sendStickyBroadcast().

Tipe-tipe *Intent*
Android memiliki dua tipe intent, yaitu

1. *Intent* eksplisit.

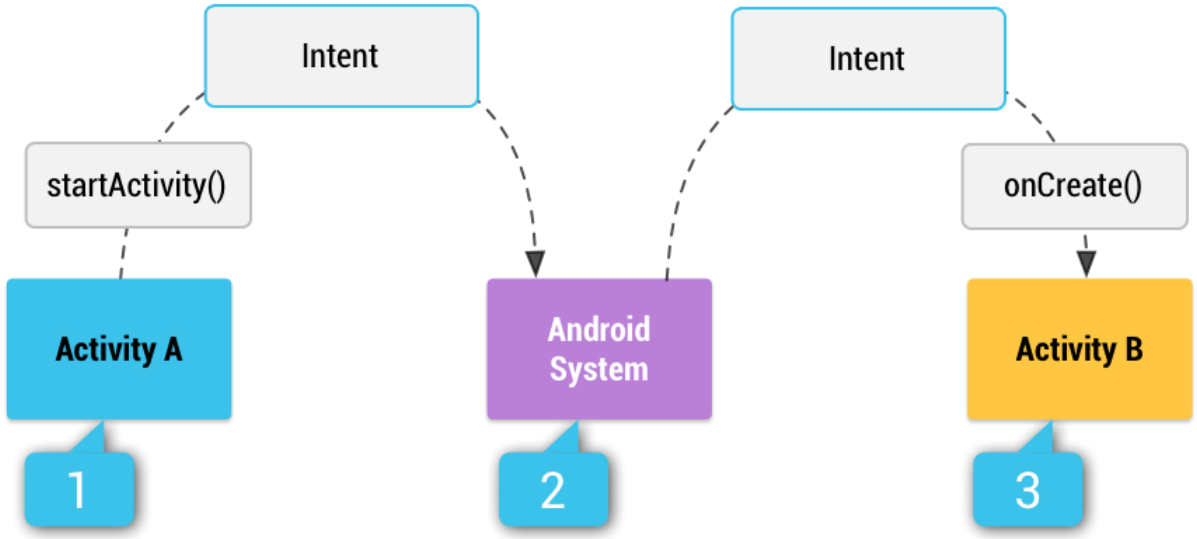
Maksud *intent* eksplisit adalah untuk memulai komponen pada aplikasi yang sama.

2. *Intent* implisit

Maksud *intent* yang implisit adalah untuk memulai komponen yang berada diluar aplikasi yang sedang digunakan. Misalnya, jika anda ingin membuka aplikasi map saat sedang berada di aplikasi *browser*.

Gambar dibawah menunjukkan bagaimana *intent* implisit digunakan untuk memulai aktivitas lain

1. Misalkan AktivitasA mengirim *intent* berisi target komponen yang ingin dicapai dengan menjalankan metode startActivity().
2. Sistem Android akan menerima *intent* tersebut, memfilter semua aplikasi terdaftar agar mendapatkan target yang dimaksud oleh *intent* tersebut.
3. Ketika kecocokan ditemukan, sistem akan meneruskan *intent* pada memulai aktivitas target (Aktivitas B).

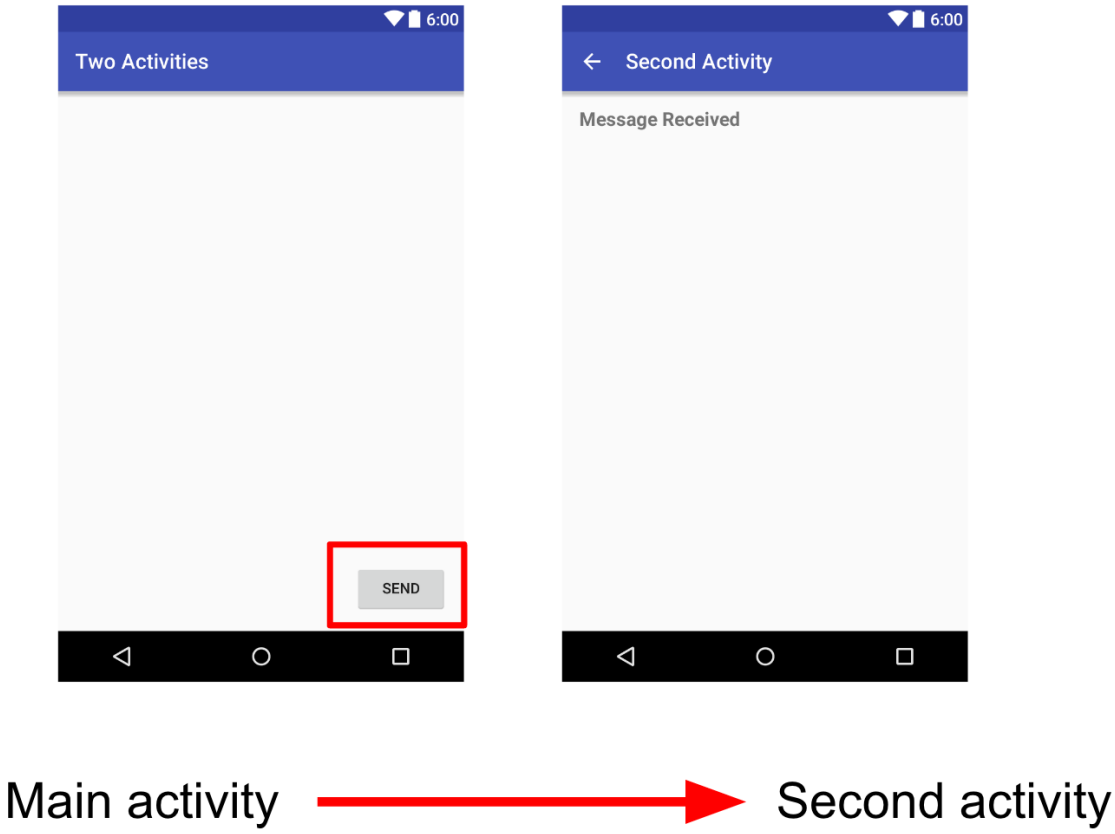


5. Latihan Aplikasi DuaAktifitas

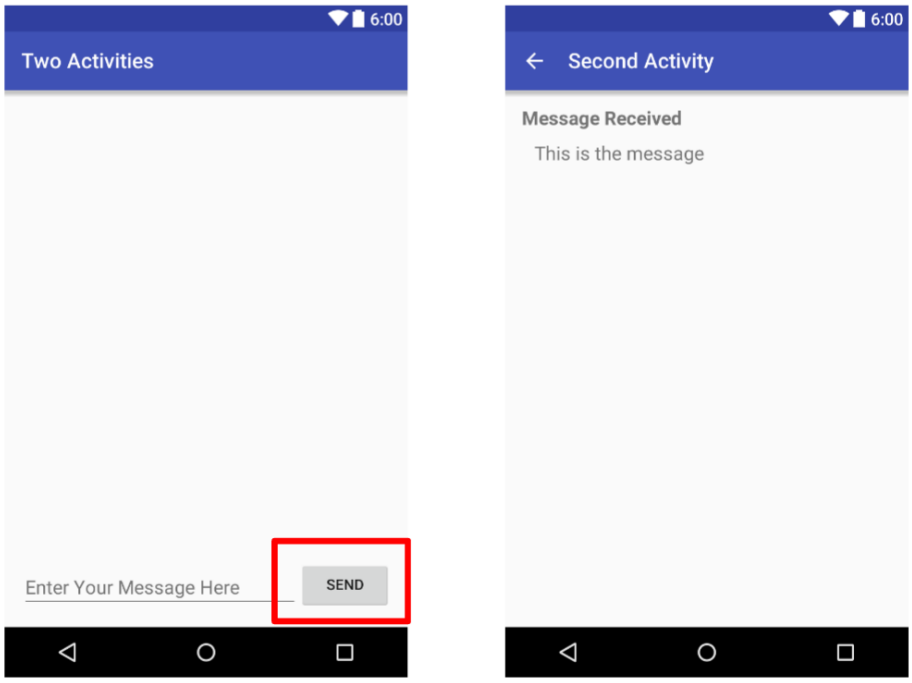
Pada latihan ini anda akan belajar untuk membuat sebuah aplikasi yang memiliki dua aktivitas. Lebih daripada itu, anda juga akan mempelajari teknik untuk mengirimkan data dari aktifitas utama ke aktifitas kedua menggunakan *intent* yang nantinya data tersebut akan ditampilkan pada aktifitas yang kedua.

Aplikasi ini akan dibuat berdasarkan dua tahapan berikut:

Pada tahap pertama, anda akan membuat aplikasi yang aktivitas utamanya berisi satu tombol, SEND. Ketika pengguna mengklik tombol ini, aktivitas utama menggunakan *intent* untuk memulai aktivitas kedua.

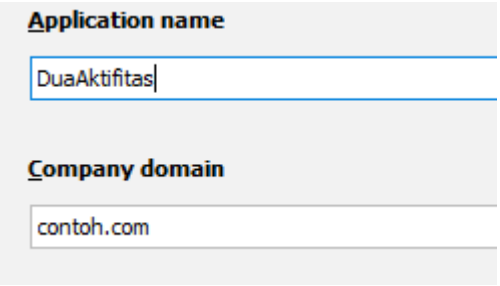


Pada tahap kedua, Anda menambahkan tampilan *EditText* ke aktivitas utama. User akan memasukkan pesan dan klik SEND. Aktivitas utama menggunakan *intent* untuk memulai aktivitas kedua dan mengirim pesan yang diketik *user* kepada aktivitas kedua. Nantinya, aktivitas kedua menampilkan pesan yang diterima seperti pada gambar dibawah ini.



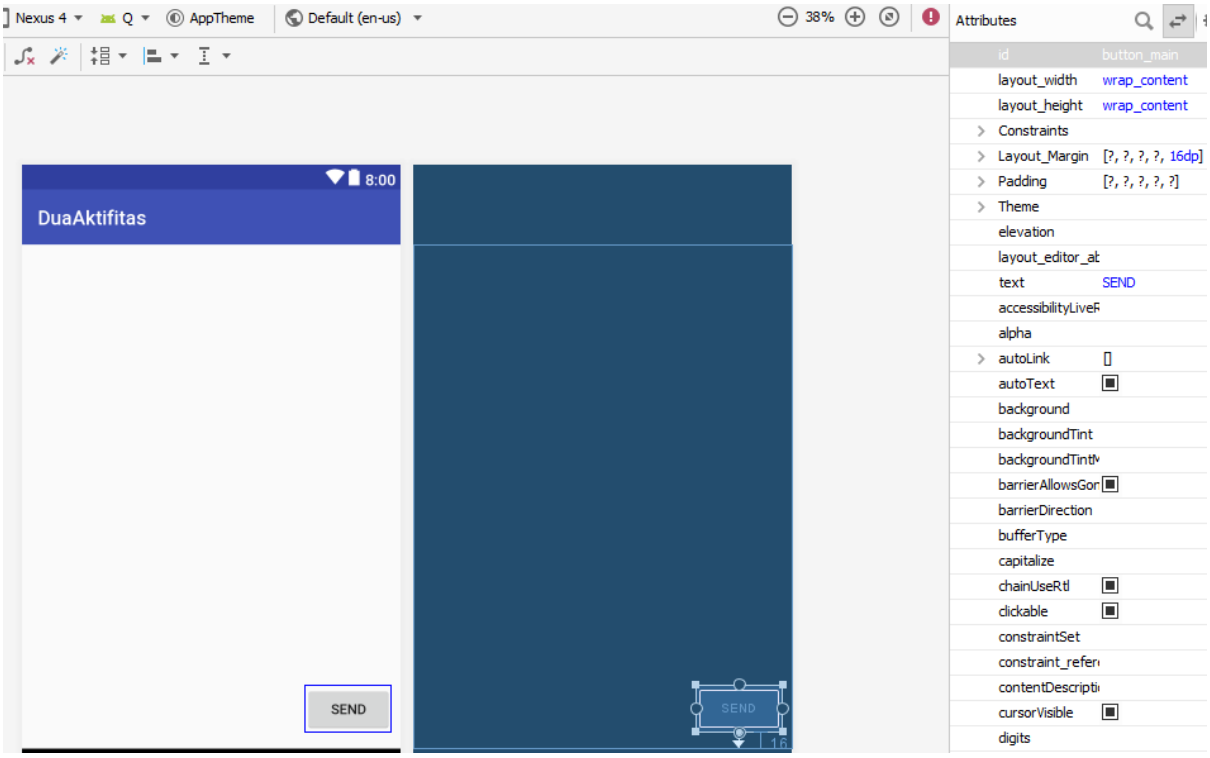
5.1 Membuat Proyek Aplikasi DuaAktifitas

Buatlah sebuah proyek pada IDE Android Studio dan beri nama proyek DuaAktifitas dengan nama company domain contoh.com.



Gunakan *Empty Activity* dan klik *Next*. Biarkan setingan *default* seperti pada modul sebelumnya dan klik *Finish*.

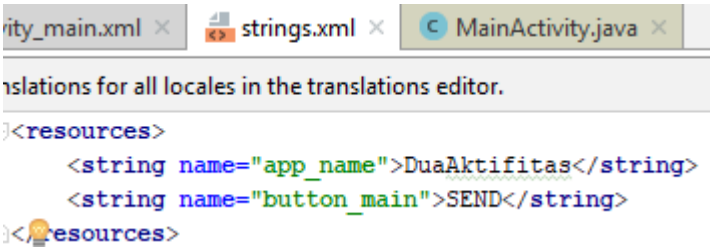
Buatlah file activity_main.xml dan pilih tab *Design* kemudian *delete* TextView (yang berisi nilai *Hello World*). Pastikan setingan *autoconnect* telah dipilih untuk memberikan *constraint*, kemudian tariklah sebuah *Button* dari *palette* kepada sisi kanan bawah layout tersebut. Pada bagian atribut di sebelah kanan IDE, set nilai atribut ID menjadi *button_main* dan pastikan nilai atribut *layout_width* dan *layout_height* adalah *wrap_content*.



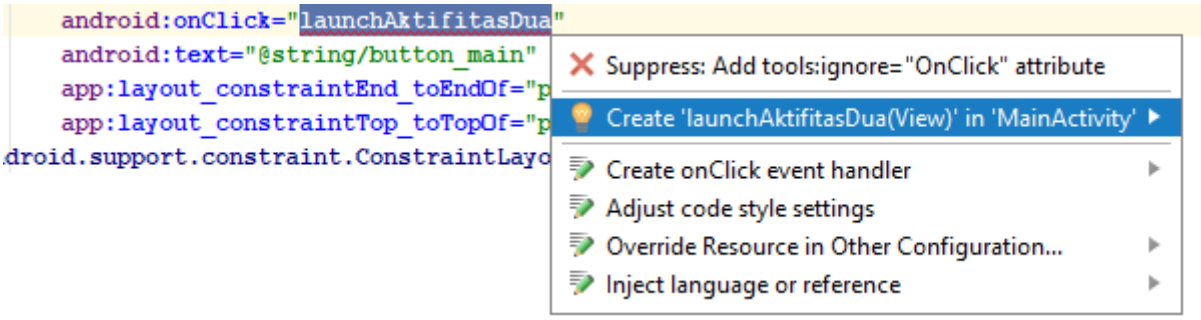
Sekarang pilih tab *Text* untuk mengedit file activity_main.xml tersebut. Tambahkan atribut dibawah ini pada *Button*.

```
android:onClick="launchAktifitasDua"
android:text="@string/button_main"
```

Note: pastikan string button_main dideklarasikan di file strings.xml seperti berikut



Berikutnya, anda harus mengimplementasikan metode launchAktifitasDua tersebut dengan membloknya dengan kursor kemudian ketik Alt+Enter dan pilih 'launchSecondActivity(View)' di 'MainActivity seperti gambar berikut.

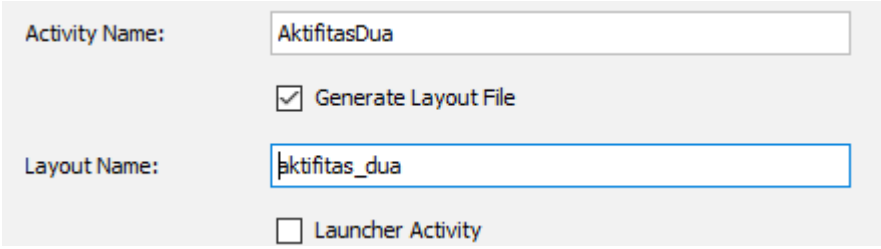


Setelah itu, file MainActivity.java akan terbuka dan IDE akan membuat metode launchSecondActivity() secara otomatis.

5.2 Membuat dan Memanggil Aktifitas Kedua

Berikutnya, anda akan menambahkan aktivitas kedua pada aplikasi. Anda akan memodifikasi file AndroidManifest.xml untuk mendefinisikan aktivitas utama sebagai induk dari aktivitas kedua. Kemudian anda akan memodifikasi metode launchAktifitasDua() yang sudah dibuat pada aktifitas utama diatas dengan mendeklarasikan *intent* untuk memicu aktivitas kedua saat Anda mengklik tombol SEND.

Sekarang buatlah aktifitas kedua. Klik folder app kemudian pilih **File** > **New** > **Activity** > **Empty Activity**. Beri nama aktifitas tersebut sebagai AktifitasDua dan nama layout aktifitas_dua kemudian klik Finish.



Dengan demikian, IDE akan menambahkan layout aktifitas_dua.xml dan file java AktifitasDua.java kepada folder app.

Selanjutnya buka **manifests > AndroidManifest.xml** dan cari elemen

```
<activity android:name=".AktifitasDua"></activity>
```

Kemudian ubah elemen tersebut menjadi berikut

```
<activity android:name=".AktifitasDua"
    android:label="@string/nama_aktifitas2"
    android:parentActivityName=".MainActivity">
</activity>
```

Atribut label akan menambahkan judul pada bar app di AktifitasDua. Note: jangan lupa tambahkan string nama_aktifitas2 di strings.xml yaitu

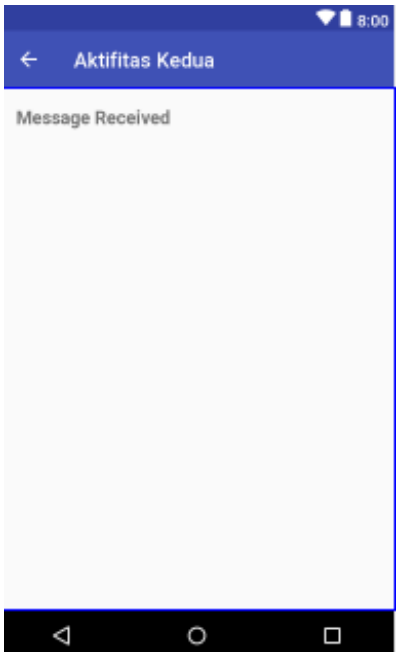
```
<resources>
    <string name="app_name">DuaAktifitas</string>
    <string name="button_main">SEND</string>
    <string name="nama_aktifitas2">Aktifitas Kedua</string>
</resources>
```

Atribut android:parentActivityName mengindikasikan bahwa MainActivity adalah induk dari AktifitasDua. Pada aplikasi, setingan ini akan menghasilkan panah yang menghadap kekiri sehingga *user* dapat kembali ke MainAcitivity.

Sekarang, kita akan menambahkan *TextView* pada layout aktifitas_dua.xml. Buka file xml tersebut dan klik tab Design. Tariklah sebuah *TextView* dari *palette* dan letakkan pada sudut kiri-atas layout. Lalu set atribut sesuai nilai-nilai berikut ini

Attribute	Value
id	text_header
Top margin	16
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	Message Received
textAppearance	AppCompat.Medium
textStyle	B (bold)

Nilai atribut *textAppearance* tersebut akan memberikan gaya tulisan tema spesial Android (pembahasan khusus mengenai tema/*style* akan anda dapatkan pada module sesi 6). Layout akan menjadi seperti ini



Klik tab *Text* lalu edit file aktifitas_dua.xml dan pindahkan text “Message Received” pada file strings.xml dengan nama text_header. Kodenya akan menjadi seperti berikut

```
<TextView
    android:id="@+id/text_header"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:text="@string/text_header"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Layout ini akan menerima pesan yang dikirim dari layout utama.

Sekarang, anda bisa menambahkan *intent* eksplisit pada tombol SEND yang ada pada layout aktifitas utama. Pertama-tama, buka file MainActivity.java. Lalu, buat *intent* baru pada metode launchAktifitasDua(). Konstruktor intent eksplisit memiliki dua parameter yaitu *context* (gunakan *this* sebagai *context*) dan komponen yang dituju.

```
public void launchAktifitasDua(android.view.View view) {
        Intent intent = new Intent( packageContext: this, AktifitasDua.class);
        startActivity(intent);
    }
```

Note: pastikan import android.content.Intent; sudah tersedia pada bagian paling atas (jika belum maka anda harus tambahkan, kadang ketika memori kurang anda perlu menambahkan setingan berikut secara manual).

```
package com.contoh.duaaktifitas;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

Sekarang coba jalankan aplikasi (dengan menekan tombol panah hijau). Jika anda menekan tombol SEND maka *MainActivity* akan mengirim *intent* pada sistem Android untuk memicu Aktifitas Kedua. Untuk kembali ke aktifitas utama, klik panah kekiri pada bar app (bagian atas aplikasi) atau tombol kembali.

5.3 Mengirim Pesan dari Aktifitas Utama ke Aktifitas Kedua

Selain berpindah dari aktifitas utama ke aktifitas kedua menggunakan *intent*, anda juga dapat menggunakan *intent* untuk mengirim data antar aktifitas. Terdapat dua cara untuk melakukan hal ini

- Menggunakan data field. Data *intent* adalah URI yang mengidentifikasikan data spesifik. Jika data yang ingin dikirimkan bukanlah URI ataupun data yang jumlahnya banyak, maka anda dapat menggunakan *intent* ekstra.

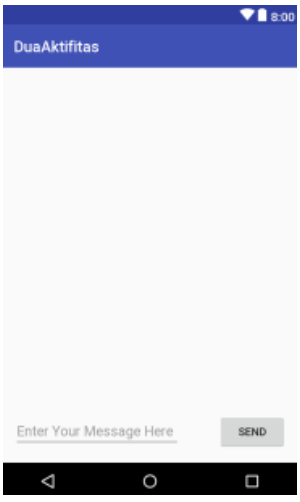
- Menggunakan *intent* ekstra. *Intent* ekstra adalah pasangan *key/value* yang disimpan didalam *Bundle* (koleksi dari pasangan *key/value*).

Sekarang anda akan memodifikasi *intent* eksplisit yang telah dibuat pada MainActivity untuk menambahkan data menggunakan *Bundel*. Kemudian, anda akan memodifikasi AktifitasDua untuk menerim data tersebut (dari *Bundle*) lalu menampilkannya di layout aktifitas kedua.

Buka **activity_main.xml**, gunakan tab *Design* lalu tarik *EditText* dari *palette* dan letakkan pada bagian bawah layout. Kemudian, set atribut *EditText* tersebut seperti berikut ini

Attribute	Value
id	editText_main
Right margin	8
Left margin	8
Bottom margin	16
layout_width	match_constraint
layout_height	wrap_content
inputType	textLongMessage
hint	Enter Your Message Here
text	(Delete any text in this field)

Layout utama akan menjadi berikut



Kemudian klik tab *Text* untuk mengedit file XML aktifitas utama, kemudian buatlah ”Enter Your Message Here” menjadi string global di strings.xml (ganti dengan nama editText_main) seperti yang telah dijelaskan sebelumnya (biasakan gunakan string global agar string tersebut dapat digunakan juga pada komponen/elemen aplikasi lainnya).

Lalu, bukalah file MainActivity.java dan deklarasikan variabel EXTRA_MESSAGE pada bagian atas dari kelas. Kemudian tambahkan variabel *EditText*, metode findViewById() dan getText() untuk mendapatkan string yang dimasukkan user, kemudian gunakan putExtra untuk memindahkan string tersebut kepada EXTRA_MESSAGE. Sekarang kode pada MainActivity akan menjadi sepert dibawah ini

```
package com.contoh.duaaktifitas;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    public static final String EXTRA_MESSAGE = "com.contoh.android.duaaktifitas.extra.MESSAGE";
    private EditText mMessageEditText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mMessageEditText = findViewById(R.id.editText_main);
    }
public void launchAktifitasDua(android.view.View view) {
        Intent intent = new Intent( packageContext: this, AktifitasDua.class);
        String message = mMessageEditText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE,message);
        startActivity(intent);
    }
}
```

Kemudian sekarang tambahkan *TextView* pada aktifitas kedua untuk menerima dan menampilkan pesan yang akan dikirimkan dari aktifitas utama.

Buka aktifitas_dua.xml, kemudian tarik *TextView* pada bagian bawah layout, kemudian set nilai-nilai atribut berikut ini

Attribute	Value
id	text_message
Top margin	8
Left margin	8
layout_width	wrap_content
layout_height	wrap_content
text	(Delete any text in this field)
textAppearance	AppCompat.Medium

Kode xml tersebut akan menjadi seperti berikut ini

```
<TextView
    android:id="@+id/text_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginStart="16dp"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    tools:layout_marginLeft="8dp"
    tools:layout_marginTop="8dp" />
```

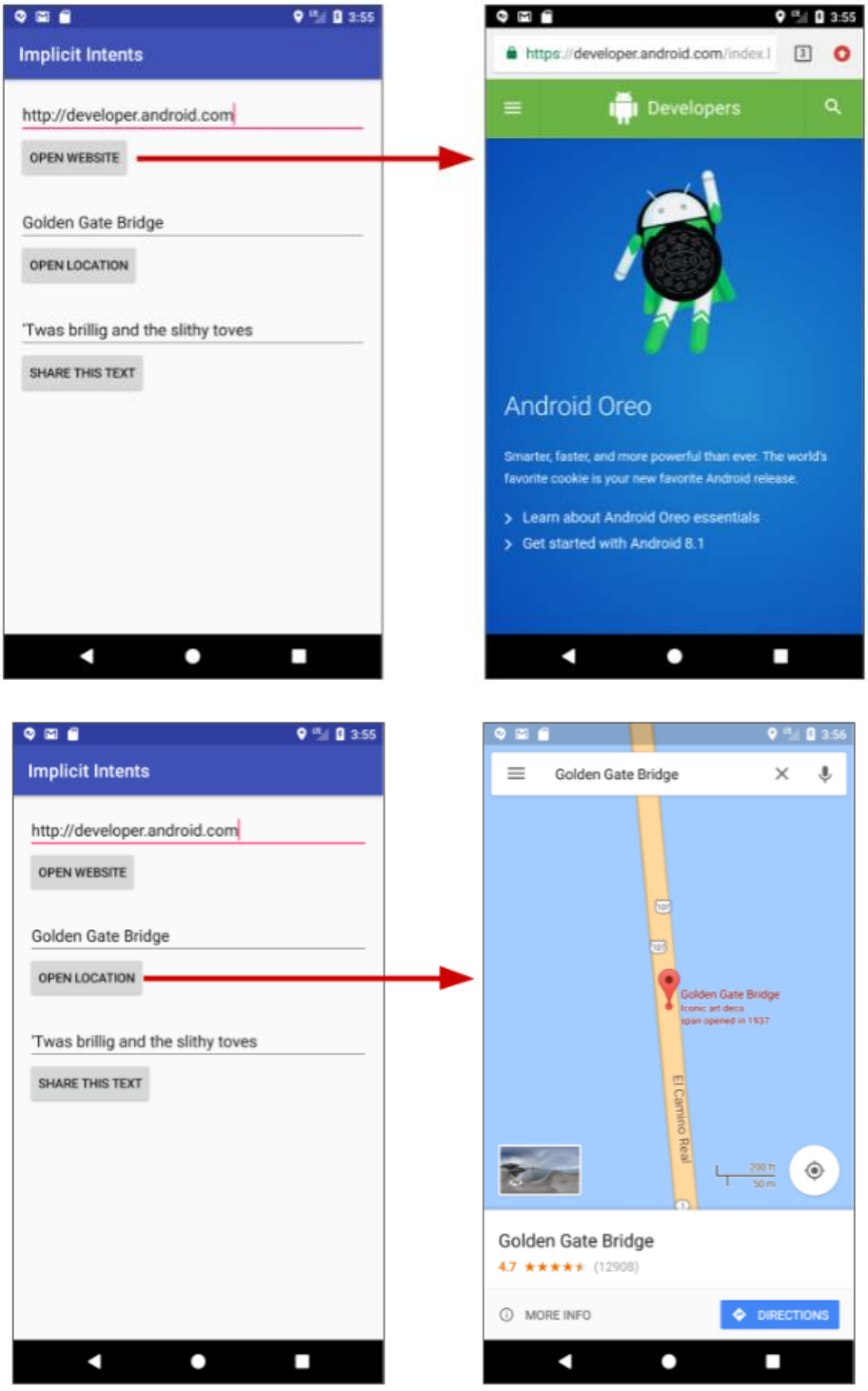
Namun, layout nampak sama saja seperti sebelum ditambahkan *TextView* karena *View* tersebut masin belum mengandung suatu text/mendapatkan pesan.

Kemudian bukalah file AktifitasDua.java. Pada metode onCreate(), buatlah objek *intent*. Kemudian gunakan getStringExtra ntuk mendapatkan nilai pada variabel EXTRA_MESSAGE pada *MainActivity*. Lalu, gunakan findViewById() untuk mendapatkan referensi *TextView* text_message. Akhirnya, set *text* yang didapatkan dari *intent* extra pada *TextView* tersebut. Kode pada metode onCreate() tersebut harusnya berbentuk seperti dibawah ini

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.aktifitas_dua);
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
    TextView textView = findViewById(R.id.text_message);
    textView.setText(message);
}
```

Jalankan aplikasi, ketika anda mengisi suatu pesan pada *editText* yang ada di MainActivity dan mengklik tombol SEND maka aktifitas kedua akan muncul dan menampilkan pesan tersebut.

Demikianlah anda telah membuat aplikasi untuk berpindah antara dua aktifitas menggunakan *intent*. Aplikasi tersebut hanya berfokus pada penggunaan *intent* eksplisit. Namun terdapat juga *intent* implisit agar aktifitas suatu aplikasi dapat memanggil aktifitas yang berada pada aplikasi lainnya seperti dua contoh dibawah ini.



Pada kedua penerapan *intent* implisit diatas, ketika tombol OPEN WEBSITE diklik, maka aktifitas yang sedang berjalan akan memicu sistem Android untuk membuka browser dan menampilkan website sesuai alamat yang dimasukkan oleh *user*. Sedangkan ketika tombol OPEN LOCATION diklik maka aktifitas akan meluncurkan aktifitas map sesuai dengan lokasi yang dimasukkan oleh *user*.

Silahkan anda belajar lebih lanjut mengenai *intent* implisit diatas secara mandiri di https://codelabs.developers.google.com/codelabs/android-training-activity-with-implicit-intent/index.html?index=..%2F..android-training#5.

Simpulan

Sesi ini telah menjelaskan komponen aplikasi yang paling mendasar yaitu aktifitas yang biasanya memiliki sebuah layout/UI. Sesi ini juga telah memperkenalkan objek *intent* yang digunakan sebagai utusan untuk mengirim data dari sebuah aktifitas kepada aktifitas lainnya. Pada sesi berikutnya, anda akan mempelajari bagaimana membuat menu agar tampilan aplikasi anda semakin standar dan menarik.

Referensi
https://codelabs.developers.google.com/codelabs/android-training-create-an-activity/index.html?index=../..android-training#5