Table of Contents

Materi

Lab

last modified by arrizaqu , 2018/01/19 14:12:26

Mengapa Harus Android ?

Di zaman modern ini, istilah Android sudah menjadi hal yang biasa di tengah masyarakat sekarang, karena dimana pun kita selalu dihadapkan dengan koneksi yang tiada henti dengan smartphone dan selalu tidak terlepaskan dengan aplikasi mobile baik aplikasi hiburan, transaksi jual beli, banking atau aplikasi lainnya, dimana aplikasi tersebut sangat membantu aktivitas keseharian, terkhusus adalah Android Smartphone yang memang di masyarakat terutama di Indonesia sangat banyak penggunanya.

Android Version

Seperti halnya dalam penamaan dan versi Android yang di assign oleh google yang diketahui adalah terinpirasi dari nama kue yang terlihat manis jika dimengkonsumsinya, berikut ini adalah nama dan versi yang sudah beredar dari munculnya Android OS yang pernah ada, diantaranya :

Oreo

Nougat: Versions 7.0-

Marshmallow: Versions 6.0 -

Lollipop: Versions 5.0 -

Kit Kat: Versions 4.4-4.4.4; 4.4W-4.4W.2

Jelly Bean: Versions 4.1-4.3.1

Ice Cream Sandwich: Versions 4.0-4.04

Versi Diatas adalah versi standart jika ingin membuat aplikasi Android, paling tidak ICS adalah versi yang paling dasar harus di setup.

Referensi

http://socialcompare.com/en/comparison/android-versions-comparison

https://www.android.com/

last modified by arrizaqu , 2018/01/19 14:30:22

Activity

Secara Default Activity Class telah disediakan oleh Android. Activity merupakan Class yang dirancang untuk meng-handle atau berperan untuk menangani User Interface (UI) ketika UI tersebut dapat berinteraksi dengan Pengguna. untuk mengimplentasikan Activity dalam aplikasi yang sedang dibuat, maka Class tersebut harus meng-extends Activity class, seperti berikut :

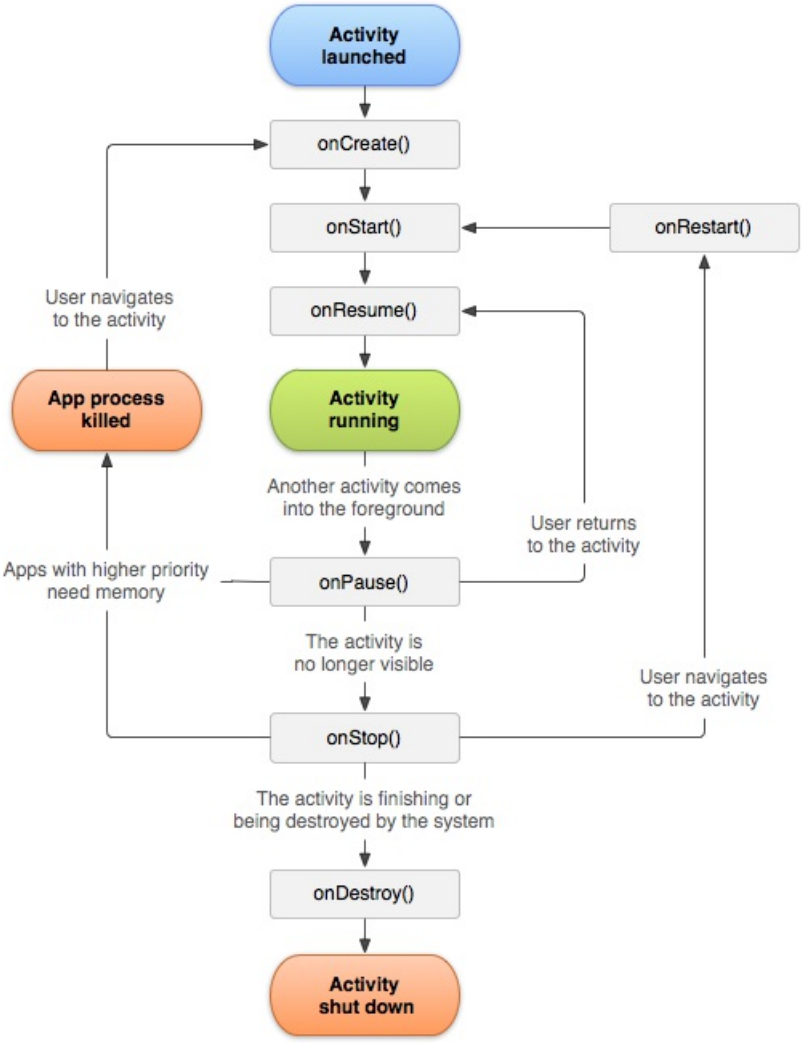
```
public class MainActivity extends Activity {

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

Life Cycle Activity

Android Activity memiliki method - method dasar yang akan dijalankan ketika sebuah Activity dijalankan, sehingga ketika program harus memodifikasi method - method tersebut maka developer harus meng-override method tersebut, diantara method - method Activity yang menjadi Life Cycle Activity android, diantaranya adalah :

1. *onCreate()*
2. *onStart()*
3. *onResume()*
4. *onPause()*
5. *onRestart()*
6. *onStop()*
7. *onDestroy()*

Untuk siklus dari method Android Activity diatas dapat digambarkan sebagai berikut dibawah ini :



Registrasi Activity

Dalam membuat Activity baru hal yang tidak boleh lupa adalah mendaftarkan Java Class yang telah di set sebagai Activity di dalam File Manifest, seperti berikut :

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

Referensi

https://developer.android.com/reference/android/app/Activity.html

last modified by masyda arrizaqu , 2018/01/23 09:39:46

Retrofit

Merupakan Library yang biasa digunakan untuk push atau get dari Request HTTP API,

Pembahasan

1. Instalasi Retrofit
2. Type Data Converter yang di support
3. Mendefinisikan RequestInterfaceMethod
4. GET Example
5. POST Example
6. PUT Example
7. Mengirimkan Gambar atau File
8. Issue

Instalasi Retrofit

Untuk Praktek pada pembahasan membutuhkan 3 library dalam penggunaan get data json menggunakan Retrofit, yaitu :

1. Retrofit
2. Converter Gson
3. OkHttp

Maven

```
<dependency>
  <groupId>com.squareup.retrofit2</groupId>
  <artifactId>retrofit</artifactId>
  <version>2.3.0</version>
</dependency>
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.squareup.okhttp/okhttp -->
<dependency>
    <groupId>com.squareup.okhttp</groupId>
    <artifactId>okhttp</artifactId>
    <version>2.7.5</version>
</dependency>
```

Gradle

```
compile 'com.squareup.retrofit2:retrofit:2.3.0'
compile 'com.squareup.retrofit2:converter-gson:2.3.0'
compile 'com.squareup.okhttp3:okhttp:3.6.0'
```

Type Data Converter yang di Support

Berikut ini adalah library yang di support dalam menggunakan Retrofit, diantaranya adalah :

- Gson : `com.squareup.retrofit2:converter-gson`
- Jackson : `com.squareup.retrofit2:converter-jackson`
- Moshi : `com.squareup.retrofit2:converter-moshi`
- Protobuf : `com.squareup.retrofit2:converter-protobuf`
- Wire : `com.squareup.retrofit2:converter-wire`
- Simple XML : `com.squareup.retrofit2:converter-simplexml`
- Scalars (primitives, boxed, and String): `com.squareup.retrofit2:converter-scalars`

Mendefinisikan RequestInterfaceMethod

Memahami untuk InterfaceEndpoint API seperti contoh script diberikut :

```
public interface EndpointInterface {
    // Request method and URL specified in the annotation
    // Callback for the parsed response is the last parameter

    @GET("users/{username}")
    Call<User> getUser(@Path("username") String username);

    @GET("group/{id}/users")
    Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);

    @POST("users/new")
    Call<User> createUser(@Body User user);
}
```

dari contoh script diatas ada beberapa Annotasi yang penting di ketahui, sebagai berikut :

Annotation	Description
`@Path`	variable substitution for the API endpoint (i.e. username will be swapped for `{username}` in the URL endpoint).
`@Query`	specifies the query key name with the value of the annotated parameter.
`@Body`	payload for the POST call (serialized from a Java object to a JSON string)
`@Header`	specifies the header with the value of the annotated parameter

GET Example

```
public void getData(){
        Gson gson = new GsonBuilder()
                .setDateFormat("yyyy-MM-dd'T'HH:mm:ssZ")
                .create();

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("https://batch131.herokuapp.com/")
                .addConverterFactory(GsonConverterFactory.create(gson))
                .build();

        final InterfaceRequest requests = retrofit.create(InterfaceRequest.class);
        Call<List<MateriEntity>> mycall = requests.materies();
        mycall.enqueue(new Callback<List<MateriEntity>>() {
            @Override
            public void onResponse(Call<List<MateriEntity>> call, Response<List<MateriEntity>> response) {
                int code = response.code();
                List<MateriEntity> listMateri = response.body();
                for(MateriEntity entity : listMateri){
                    Log.d("result : ", "idTrayek : "+ entity.getIdTrayek());
                }
            }

            @Override
            public void onFailure(Call<List<MateriEntity>> call, Throwable t) {

            }
        });
    }
```

Issue

1. GsonConverterFactory Cannot be resolve (Solved) : https://stackoverflow.com/questions/33304420/retrofit-2-example-tutorial-but-gsonconverterfactory-display-error-cannot-resol, (Library ini dipakai untuk mengconvert Json Format ke Object di dalam java object)
2. Unable to create converter for java.util.List Retrofit: https://stackoverflow.com/questions/34315499/unable-to-create-converter-for-java-util-list-retrofit-2-0-0-beta2 (secara simple dengan error ini membutuhkan 3 library yang harus ada dalam penggunaan retrofit2 : retrofit, okhttp, dan json converter)
3. Best Praktis More Retrofit Example : https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit

Referensi

http://square.github.io/retrofit/

https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit

last modified by arrizaqu , 2018/01/22 22:44:16

Issue

1. GsonConverterFactory Cannot be resolve (Solved) : https://stackoverflow.com/questions/33304420/retrofit-2-example-tutorial-but-gsonconverterfactory-display-error-cannot-resol
2. Unable to create converter for java.util.List Retrofit: https://stackoverflow.com/questions/34315499/unable-to-create-converter-for-java-util-list-retrofit-2-0-0-beta2

last modified by arrizaqu , 2018/01/19 15:02:44

hallo ini nantinya adalah lab untuk pembahasan Activity ya..

last modified by arrizaqu , 2018/01/22 22:44:16

Lab Retrofit

1. Main Activity
2. POJO Example
3. EndPoint API
4. Post Example

MainActivity

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //gson format
        Gson gson = new GsonBuilder()
                .setDateFormat("yyyy-MM-dd'T'HH:mm:ssZ")
                .create();

        Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("https://batch131.herokuapp.com/")
                .addConverterFactory(GsonConverterFactory.create(gson))
                .build();

       final InterfaceRequest requests = retrofit.create(InterfaceRequest.class);
       Call<List<MateriEntity>> mycall = requests.materies();
       mycall.enqueue(new Callback<List<MateriEntity>>() {
           @Override
           public void onResponse(Call<List<MateriEntity>> call, Response<List<MateriEntity>> response) {
               int code = response.code();
               List<MateriEntity> listMateri = response.body();
               for(MateriEntity entity : listMateri){
                   Log.d("result : ", "idTrayek : "+ entity.getIdTrayek());
               }
           }

           @Override
           public void onFailure(Call<List<MateriEntity>> call, Throwable t) {

           }
       });
    }
}
```

POJO Example

```
public class MateriEntity {

    String jamPesan;
    String confirmJamPesan;
    String statusConfirm;
    String atasNamaPemabayaran;
    int idTrayek;
    int idPemesan;
```

```
    ... setter and getter
}
```

EndPoint API

```
public interface InterfaceRequest {
    @GET("kelompok3-ws/list")
    Call<List<MateriEntity>> materies();

    @POST("kelompok3-ws/save")
    Call<RequestBody> save(@Body RequestBody materiEntity);

}
```

Post Example

- Define JSON TYPE variable
- Prepare Json Object Data
- Execution Function

Define JSON TYPE variable

```
 public static final MediaType JSON = MediaType.parse("application/json; charset=utf-8");
```

Prepare Json Object Data

```
try{
    JSONObject jsonObject = new JSONObject();
    jsonObject.put("codePesan", "bsik123");
    jsonObject.put("idTrayek", 56789);
   // jsonObject.put("statusConfirm", "pending");
    exampleInsertData(jsonObject);
} catch(Exception e){
    e.printStackTrace();
}
```

Execution Function

```
public void exampleInsertData(JSONObject entity) throws IOException {
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://batch131.herokuapp.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    final InterfaceRequest requests = retrofit.create(InterfaceRequest.class);
    Log.d("ss", entity.toString());
    RequestBody requestBody = RequestBody.create(JSON, entity.toString());
    Call<RequestBody> call = requests.save(requestBody);
    call.enqueue(new Callback<RequestBody>() {

        @Override
        public void onResponse(Call<RequestBody> call, Response<RequestBody> response) {

        }

        @Override
        public void onFailure(Call<RequestBody> call, Throwable t) {

        }
    });
}
```