

Modul Praktikum Flutter

Bagian Tiga: Koneksi Web Service



Yudi Wibisono (yudi@upi.edu)
Ilmu Komputer, Universitas Pendidikan Indonesia (cs.upi.edu)

versi: Beta Mei 2023



<http://creativecommons.org/licenses/by-nc-sa/3.0/>

Modul ini bebas di-copy, didistribusikan, ditransmit dan diadaptasi/modifikasi/diremix dengan syarat tidak untuk komersial, pembuat asal tetap dicantumkan dan hasil modifikasi dishare dengan lisensi yang sama.

Modul ini adalah lanjutan modul bagian kedua (UI):

<https://docs.google.com/document/d/1W8OEVGdNng0nMSQVCvCea3WSYLacMpdoagub77s7IYI/edit?usp=sharing>

Daftar Isi

Daftar Isi	2
Koneksi dengan Web Service	4
JSON & Serialization	4
Package http dan widget FutureBuilder	6
Mengambil data: GET	7
Insert data: POST	11
Update seluruh data: PUT	14
Update sebagian data: PATCH	15
Delete data: DELETE	16
ListView Builder dan FutureBuilder	18

Koneksi dengan Web Service

JSON & Serialization

JSON adalah format standar yang saat ini umum digunakan untuk pertukaran data antar sistem, khususnya web service. JSON (JavaScript Object Notation) populer karena formatnya mudah dibaca baik oleh mesin maupun manusia.

Contoh JSON:

```
{"nama": "Budi Martami", "bahasa": ["C++", "Python"]}
```

Contoh JSON yang lebih kompleks:

```
{
  "nama": "Budi Martami",
  "alamat": "Ujung Berung",
  "hobi": ["membaca", "berlari"],
  "anak": [
    {
      "nama": "Ahmad Aulia",
      "tgl_lahir": "01-01-2010"
    },
    {
      "nama": "Sandra Permana",
      "tgl_lahir": "04-07-2014"
    }
  ]
}
```

Selain string, JSON juga dapat menggunakan tipe lain seperti integer, float dan boolean.

Contoh:

```
{
  "nama": "Budi",
  "bahasa_asing": ["Inggris", "Perancis"],
  "menikah": true,
  "tinggi_badan": 170,
  "ipk": 3.75
}
```

Dapat dilihat format JSON mirip dengan struktur key-value yang ditulis sebagai

"key" : "value", dan struktur array dengan simbol []. Pasangan key-value dan array dapat digabung dalam satu kelompok menggunakan kurung kurawal { }. Struktur ini dapat dibuat bersarang, misalnya ada array yang didalamnya ada kelompok yang mengandung array dan seterusnya.

Pada Dart, pemrosesan JSON dapat menggunakan method `jsonencode()` dari package `dart:convert`.

Contoh penggunaannya:

```
import 'dart:convert';

void main() {
  String jsonString =
    '{"nama": "Budi Martami", "umur":17, "list_bahasa": ["C++", "Python"]}';

  Map<String, dynamic> mhsJson = jsonDecode(jsonString); //json ke Map

  print("nama: ${mhsJson['nama']}");
  print("umur: ${mhsJson['umur']}");
  print("skill: ${mhsJson['list_bahasa']}");

  //cetak satu-satu
  for (String val in mhsJson['list_bahasa']) {
    print(val);
  }
}
```

Sedangkan untuk encodenya (mengubah dari string ke JSON):

```
String nama = "Ahmad Aulia";
int umur = 20;
List<dynamic> listBahasa = ["php", "js"];

String mhs2json =
  jsonEncode({"nama": nama, "umur": umur, "list_bahasa": listBahasa});

print(mhs2json);
```

Latihan 1

Buatlah JSON untuk transkrip mahasiswa. Lalu buatlah program untuk menghitung IPK

Package http dan widget FutureBuilder

Salah satu sumber data app adalah REST API (web service). Flutter menyediakan package `http` untuk menangani hal ini.

Untuk menginstall package, edit **pubspec.yaml**, dan dibagian dependency tambahkan package http.

```
dependencies:  
  http: 1.0.0  
  flutter:  
    sdk: flutter
```

Kemudian save, Visual Studio Code akan otomatis mendownload package ini.

Catatan: nomor versi terakhir package http dapat dilihat di <https://pub.dev/packages/http/install>
Pada saat modul ini dibuat, versi terakhir adalah 1.0.0.

Untuk contoh app pada modul ini, akan digunakan API dari <https://catfact.ninja/fact> yang menampilkan fakta mengenai kucing.

Saat dicoba di browser, hasil dari API ini adalah JSON berikut (tiap dijalankan bisa berbeda):

```
{"fact": "A cat called Dusty has the known record for the most kittens.  
She had more than 420 kittens in her lifetime.", "length": 108}
```

Catatan: Berbagai sumber API gratis dapat dilihat di website:
<https://apihenry.io/free-api/>

Mengambil data: GET

Kita akan mengambil data fakta seputar kucing dari situs catfact.ninja

Pertama import package `http` dan `convert`

```
import 'package:http/http.dart' as http;
import 'dart:convert';
```

Kemudian kita perlu membuat class yang akan meng-konversi JSON dan menampung data.

```
// menampung data hasil pemanggilan API
class CatFact {
  String fakta;
  int panjang;

  CatFact({required this.fakta, required this.panjang}); //constructor

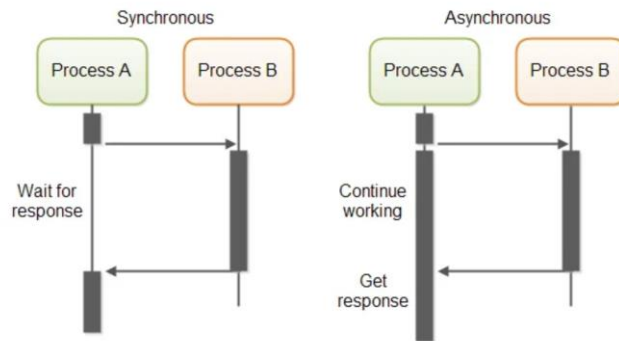
  // create objek CatFact dan isi atributnya dari json
  factory CatFact.fromJson(Map<String, dynamic> json) {
    return CatFact(
      fakta: json['fact'],
      panjang: json['length'],
    );
  }
}
```

Catatan: parameter `fromJson` bertipe `Map<String, dynamic>` karena JSON dari catfact berbentuk pasangan key value, contohnya `{"fact": "A cat called Dusty has the known record for the most kittens. She had more than 420 kittens in her lifetime.", "length": 108}`

Jika JSON berbentuk array, seperti `[{}, {}, {}]` maka tipe parameter juga perlu menyesuaikan menjadi `List<dynamic>` bukan `Map` lagi.

Selanjutnya dibagian `state` (kita menggunakan `statefull widget`), tambahkan variabel untuk menampung hasil pemanggilan API. Diperlukan class `Future` karena fungsi pemanggilan web service sifatnya asynchronous berjalan di background. Tujuannya agar aplikasi tidak freeze saat menunggu proses pemanggilan API selesai.

Perbedaan proses synchronous dengan asynchronous dapat dilihat pada gambar berikut.



Sumber:

<https://medium.com/from-the-scratch/what-is-synchronous-and-asynchronous-1a75afd039df>

```
class MyAppState extends State<MyApp> {
    late Future<CatFact> futureCatFact;
    . . .
}
```

Tambahkan method untuk mengambil data dari webservice:

```
class MyAppState extends State<MyApp> {
    late Future<CatFact> futureCatFact; //menampung hasil
    String url = "https://catfact.ninja/fact";

    //fetch data
    Future<CatFact> fetchData() async {
        final response = await http.get(Uri.parse(url));
        if (response.statusCode == 200) {
            // jika server mengembalikan 200 OK artinya berhasil,
            // parse json, return objek CatFact yang terisi
            return CatFact.fromJson(jsonDecode(response.body));
        } else {
            // jika gagal (bukan 200 OK),
            // lempar exception
            throw Exception('Gagal panggil API');
        }
    }
}
```

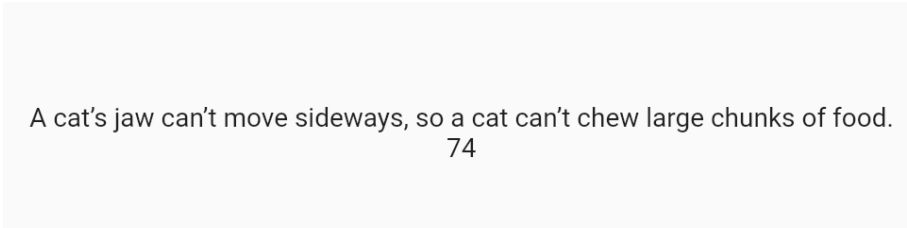
Panggil method fetch data ini saat inisiasi MyAppState: `initstate()`

```
@override
void initState() {
  super.initState();
  futureCatFact = fetchData();
}
```

Untuk menampilkan hasilnya, gunakan widget **FutureBuilder**. Widget ini otomatis akan terisi saat variabel future-nya sudah terisi secara asynchronous.

```
body: Center(
  child:
    FutureBuilder <CatFact>(
      future: futureCatFact,
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          return Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(snapshot.data!.fakta),
                Text(snapshot.data!.panjang.toString())
              ]));
        } else if (snapshot.hasError) {
          return Text('${snapshot.error}');
        }
        // default: loading spinner.
        return const CircularProgressIndicator();
      },
    ),
),
```

Jalankan dan hasilnya akan seperti ini:



A cat's jaw can't move sideways, so a cat can't chew large chunks of food.

74

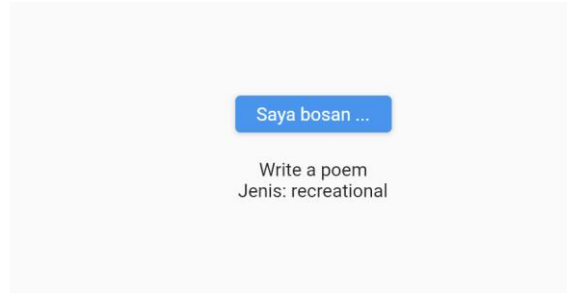
Code lengkap: <https://pastebin.com/raw/m10XqyT7>

Catatan: jika menggunakan web service lokal dan terjadi XMLHttpRequest error, ini dapat disebabkan oleh CORS (Cross-Origin Resource Sharing) karena frontend flutter app yang dijalankan di Chrome menggunakan JScript yang berkomunikasi dengan backend, sehingga backend memiliki "origin" berbeda dengan frontend. Solusinya ada di sisi server. Untuk FastAPI, solusinya adalah sbb:

```
from fastapi.middleware.cors import CORSMiddleware
from fastapi import FastAPI
app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Latihan 2

Menggunakan API pada <https://www.boredapi.com/api/activity> buatlah app yang jika button ditap akan menampilkan rekomendasi aktivitas (gambar berikut)



Kunci jawaban: <https://pastebin.com/raw/Q6ABYE0g>

Insert data: POST

Jika baru pertama kali membaca modul ini, bagian POST, PUT, PATCH, DELETE dapat dilewati karena memerlukan pembuatan web service terlebih dulu.

Catatan

Modul terkait pembuatan web service dengan FAST API:

<https://docs.google.com/document/d/1uXrfQ2E4ZudKB9Q9qgz639iJf-hiHw6URhQCEn2qXg4/edit?usp=sharing>

Kita akan menggunakan API pada modul FAST API di atas (code lengkap: <https://pastebin.com/raw/7dweXC72>)

Catatan: Flutter dan FastAPI dapat dijalankan dengan Visual Studio Code secara bersamaan di window berbeda.

Spesifikasi untuk endpoint insert data adalah sebagai berikut (**/tambah_mhs**):



Kita mengetahui proses insert berhasil jika endpoint menghasilkan nilai 201. Berbeda dengan get, untuk POST data disimpan dalam body. Header menyatakan data berbentuk JSON dan body dikirim dalam format JSON sesuai spesifikasi endpoint (code di bawah)

```
late Future<int> respPost; //201 artinya berhasil
String url = "http://127.0.0.1:8000/tambah_mhs/";
```

```

Future<int> insertData() async {
  //data disimpan di body
  final response = await http.post(
    Uri.parse(url),
    headers: <String, String>{
      'Content-Type': 'application/json; charset=UTF-8'},
    body: """
      {"nim": "13594022",
       "nama": "Sandra Permana",
       "id_prov": "12",
       "angkatan": "2020",
       "tinggi_badan": 190}
      """);
  return response.statusCode; //sukses kalau 201
}

@override
void initState() {
  super.initState();
  respPost = Future.value(0); //init
}

```

Selanjutnya untuk pemanggilan fetchData() dengan men-tap button dan menampilkan hasilnya di bawah button

```

children: [
  ElevatedButton(
    onPressed: () {
      setState(() {
        respPost = insertData(); //jika return 201 artinya sukses
      });
    },
    child: const Text('Klik Untuk Insert data (POST)'),
  ),
  Text("Hasil:"),
  FutureBuilder<int>(
    future: respPost,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        if (snapshot.data! == 201) {
          return Text("Proses Insert Berhasil!");
        }
        if (snapshot.data! == 0) {
          return Text("");
        }
      }
    },
  ),
]

```

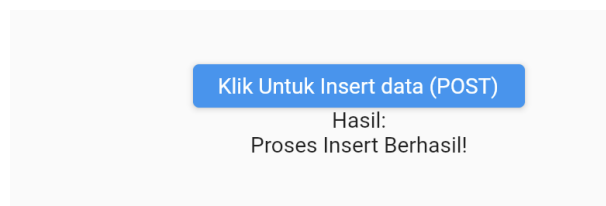
```

    } else {
        return Text("Proses insert gagal");
    }
}
// default: loading spinner.
return const CircularProgressIndicator();
})
],

```

Code lengkap: <https://pastebin.com/raw/Ne2MDp7u>

Jika berhasil:



Cek menggunakan endpoint tampil semua data untuk menampilkan data, apakah data yang baru di-insert sudah masuk.

```

[
  23,
  "13594022",
  "Sandra Permana",
  "12",
  "2020",
  190
]

```

Update seluruh data: PUT

PUT mengganti secara keseluruhan data (untuk update sebagian, gunakan PATCH)

Spesifikasi endpoint PUT untuk contoh kita adalah sebagai berikut, ada kombinasi parameter NIM dan body untuk data yang akan digantikan.

PUT /update_mhs_put/{nim} Update Mhs Put	
Parameters	
Name	Description
nim * required string (path)	<input type="text" value="nim"/>
Request body required	
Example Value Schema	
<pre>{ "nim": "string", "nama": "string", "id_prov": "string", "angkatan": "string", "tinggi_badan": 0 }</pre>	
Responses	
Code	Description
200	Successful Response

Code untuk PUT mirip dengan POST sebelumnya, perbedaannya adalah pemanggilan menggunakan `http.put` dan sukses jika return code adalah 200.

```
class MyAppState extends State<MyApp> {
  late Future<int> respPost; //201 artinya berhasil
  String url = "http://127.0.0.1:8000/update_mhs_put/";

  Future<int> fetchData() async {
    String nim = "13594022";
    //nim tambahkan di url, data tetap di body
    //pastikan http.put bukan post
    final response = await http.put(Uri.parse(url + nim),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8'
      },
      body: ""
    );
```

```

{
  "nim": "$nim",
  "nama": "Ahmad Aulia",
  "id_prov": "142",
  "angkatan": "2022",
  "tinggi_badan": 192} "");
return response.statusCode; //sukses kalau 200 }

```

Untuk memeriksa apakah berhasil, cek status code 200:

```

builder: (context, snapshot) {
  if (snapshot.hasData) {
    if (snapshot.data! == 200) {
      return Text("Proses update berhasil!");
    }
    if (snapshot.data! == 0) {
      return Text("");
    } else {
      return Text("Proses insert gagal");
    }
  }
}

```

Code lengkap:

<https://pastebin.com/raw/CMXNMqsj>

Update sebagian data: PATCH

Spesifikasi endpoint PATCH adalah sebagai berikut:

PATCH /update_mhs_patch/{nim} Update Mhs Patch	
Parameters	
Name	Description
nim * required string (path)	<input type="text" value="nim"/>
Request body required	
Example Value	Schema
<pre>{ "nama": "kosong", "id_prov": "kosong", "angkatan": "kosong", "tinggi_badan": "kosong" }</pre>	
Responses	
Code	Description
200	Successful Response

```

class MyAppState extends State<MyApp> {
  late Future<int> respPost; //201 artinya berhasil
  String url = "http://127.0.0.1:8000/update_mhs_patch/";

  Future<int> fetchData() async {
    //data disimpan di body
    String nim = "13594022";
    //nim tambahkan di url
    //pastikan http.patch
    final response = await http.patch(Uri.parse(url + nim),
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8'
      },
      body: ""
    {
      "nama": "Sandra Aulia"} ""); //hanya ganti nama saja
    return response.statusCode; //sukses kalau 200
  }
}

```

Code lengkap: <https://pastebin.com/raw/UZpPAFBc>

Delete data: DELETE

Spesifikasi endpoint delete dalam kasus kita adalah sebagai berikut

DELETE /delete_mhs/{nim} Delete Mhs	
Parameters	
Name	Description
nim * required string (path)	13594022
Responses	
Curl	
<pre>curl -X 'DELETE' \ 'http://127.0.0.1:8000/delete_mhs/13594022' \ -H 'accept: application/json'</pre>	
Request URL	
http://127.0.0.1:8000/delete_mhs/13594022	
Server response	
Code	Details
200	Response body
<pre>{ "status": "ok" }</pre>	

Codenya adalah sebagai berikut:

```

String url = "http://127.0.0.1:8000/delete_mhs/";
Future<int> fetchData() async {

```

```
String nim = "13594022";  
//nim tambahkan di url  
//pastikan http.delete  
final response = await http.delete(Uri.parse(url + nim));  
return response.statusCode; //sukses kalau 200  
}
```

Code lengkap: <https://pastebin.com/raw/9q6BN4Qu>

Pilih Image dan POST ke backend

<https://pastebin.com/raw/Zq3FSjDw>

ListView Builder dan FutureBuilder

Selanjutnya kita akan membuat contoh ListView yang diisi berdasarkan data dari web service dengan JSON yang lebih kompleks.

Jangan lupa tambahkan library http di **pubspec.yaml**

```
dependencies:  
  http: 0.13.5  
  flutter:  
    sdk: flutter
```

API yang digunakan adalah data populasi negara USA:

<https://datausa.io/api/data?drilldowns=Nation&measures=Population>

Contoh JSON yang dihasilkan adalah sebagai berikut:

```
{ "data": [  
  { "ID Nation": "01000US", "Nation": "United States", "ID  
    Year": 2019, "Year": "2019", "Population": 328239523, "Slug Nation": "united-states" },  
  
  { "ID Nation": "01000US", "Nation": "United States", "ID  
    Year": 2018, "Year": "2018", "Population": 327167439, "Slug Nation": "united-states" },  
  ...  
  
  { "ID Nation": "01000US", "Nation": "United States", "ID Year": 2013, "Year": "2013",  
    "Population": 316128839, "Slug Nation": "united-states" }  
],  
  
"source":  
[ { "measures": [ "Population" ], "annotations":  
  { ... }, "name": "acs_yg_total_population_1", "substitutions": [ ] } ] }
```

Bagian yang penting adalah atribut data yang didalamnya mengandung tahun dan jumlah populasi.

Seperti pada contoh sebelumnya, kita akan membuat class untuk menampung data JSON ini.

```
class PopulasiTahun {  
  int tahun;  
  int populasi;  
  PopulasiTahun({required this.tahun, required this.populasi});  
}  
  
class Populasi {  
  //list berisi populasi dan tahun  
  List<PopulasiTahun> ListPop = <PopulasiTahun>[];
```

```

//constructor
//ambil dari bentuk key:value dari json, isi ke array
Populasi(Map<String, dynamic> json) {
    var data = json["data"]; //bentuknya hirarkis, ambil elemen "data"
    //loop isi elemen data untuk ambil tahun dan populasi
    for (var val in data) {
        var tahun = int.parse(val["Year"]); //thn dijadikan int
        var populasi = val["Population"]; //populasi sudah int
        //tambahkan ke array
        ListPop.add(PopulasiTahun(tahun: tahun, populasi: populasi));
    }
}

//map dari json ke atribut
factory Populasi.fromJson(Map<String, dynamic> json) {
    return Populasi(json);
}
}

```

Selanjutnya fetch data dari web service saat initState().

```

//class state
class MyAppState extends State<MyApp> {
    late Future<Populasi> futurePopulasi;

    String url = "https://datausa.io/api/data?drilldowns=Nation&measures=Population";

    //fetch data
    Future<Populasi> fetchData() async {
        final response = await http.get(Uri.parse(url));
        if (response.statusCode == 200) {
            // jika server mengembalikan 200 OK (berhasil),
            // parse json
            return Populasi.fromJson(jsonDecode(response.body));
        } else {
            // jika gagal (bukan 200 OK),
            // lempar exception
            throw Exception('Gagal load');
        }
    }
}

@override
void initState() {

```

```

    super.initState();
    //ambil data dari web service
    futurePopulasi = fetchData();
  }

```

Berikutnya kita akan menampilkan data menggunakan widget FutureBuilder yang didalamnya mengandung ListView Builder

```

body: Center(
  child: FutureBuilder<Populasi>(
    future: futurePopulasi,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        return Center(
          //gunakan listview builder
          child: ListView.builder(
            itemCount: snapshot
              .data!.ListPop.length, //asumsikan data ada isi
            itemBuilder: (context, index) {
              return Container(
                decoration: BoxDecoration(border: Border.all()),
                padding: const EdgeInsets.all(14),
                child: Column(
                  mainAxisAlignment: MainAxisAlignment.center,
                  children: [
                    Text(snapshot.data!.ListPop[index].tahun
                      .toString()),
                    Text(snapshot.data!.ListPop[index].populasi
                      .toString()),
                  ]));
            }
          );
        }
      }
    ),
  ),
)

```

Hasilnya adalah sebagai berikut:

coba http	
2019	328239523
2018	327167439
2017	325719178
2016	323127515

Source code lengkap: <https://pastebin.com/raw/4hy3dukW>

Latihan 3

Menggunakan API <http://universities.hipolabs.com/search?country=Indonesia>,
buatlah ListView yang menampilkan nama dan situs universitas.