**HTTP PACKAGE JSON VE API KAVRAMLARI**

Ders 1 : Local dosyadan json veri okumak

1.Öncelikle bir proje oluşturalım.

2.main.dart içeriği aşağıdaki gibi olacaktır.

import 'package:flutter/material.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(

        primarySwatch: Colors.blue,

      ),

      home: const MyHomePage(),

    );

  }

}

class MyHomePage extends StatelessWidget {

  const MyHomePage({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Http Json'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            ElevatedButton(

                style: ElevatedButton.styleFrom(primary: Colors.green),

                onPressed: () {

                  Navigator.of(context).push(MaterialPageRoute(

                      builder: (context) => const LocalJson()));

                },

                child: const Text('Local Json')),

          ],

        ),

      ),

    );

  }

}

3.local\_json.dart isminde bir dosya tanımlanır.

import 'package:flutter/material.dart';

class LocalJson extends StatefulWidget {

  const LocalJson({Key? key}) : super(key: key);

  @override

  State<LocalJson> createState() => \_LocalJsonState();

}

class \_LocalJsonState extends State<LocalJson> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Local Json İşlemleri')),

      body: Center(),

    );

  }

4.ana dizine assets/data/arabalar.json dosyası oluşturulur.

[

    {

        "araba\_adi": "Mercedes",

        "ulke":"Almanya",

        "kurulus\_yil": 1920,

        "model": [

            {

                "model\_adi":"a serisi",

                "fiyat":15000,

                "benzinli":true

            },

            {

                "model\_adi":"c serisi",

                "fiyat":25000,

                "benzinli":false

            }

        ]

    },

    {

        "araba\_adi": "Honda",

        "ulke":"Japonya",

        "kurulus\_yil": 1960,

        "model": [

            {

                "model\_adi":"civic",

                "fiyat":12000,

                "benzinli":true

            },

            {

                "model\_adi":"jazz",

                "fiyat":10000,

                "benzinli":false

            }

        ]

    },

5.pubspec.yaml da assets içerisinde tanımlanır.

assets:

    - assets/data/arabalar.json

6.local\_json.dart içerisinde aşağıdaki metot tanımlanır.

arabalarJsonOku() async {

    String okunanString =

        await DefaultAssetBundle.of(context).loadString("data/arabalar.json");

    debugPrint(okunanString);

  }

7.local\_json içerisinde build metodunda metot çağrılır.

class \_LocalJsonState extends State<LocalJson> {

  @override

  Widget build(BuildContext context) {

    arabalarJsonOku();

    return Scaffold(

      appBar: AppBar(title: Text('Local Json İşlemleri')),

      body: Center(),

    );

  }

8.local\_json.dart da arabalarOku metoduna json dan gelen verileri çözmek ve string formatında yazdırabilmek için seçili kodlar eklenir.

arabalarJsonOku() async {

    String okunanString = await DefaultAssetBundle.of(context)

        .loadString("assets/data/arabalar.json");

    var jsonArray = jsonDecode(okunanString);

    List arabaListesi = jsonArray;

    //debugPrint(arabaListesi[1]["model"][1].toString());

9.Asıl amacımız gelen json verilerini bir model olarak oluşturup oop yapısına göre düzenlemek olmalı.Bunun için lib içerisinde model isminde bir klasör oluşturalım.Daha sonra araba\_model.dart dosyasını oluşturalım.

10.arabalar.json da aşağıdaki kısmı olduğu gibi kopyalayalım.

{

        "araba\_adi":"Mercedes",

        "ulke":"Almanya",

        "kurulus\_yil":1920,

        "model":

        [

            {

                "model\_serisi":"A serisi",

                "fiyat":1500000,

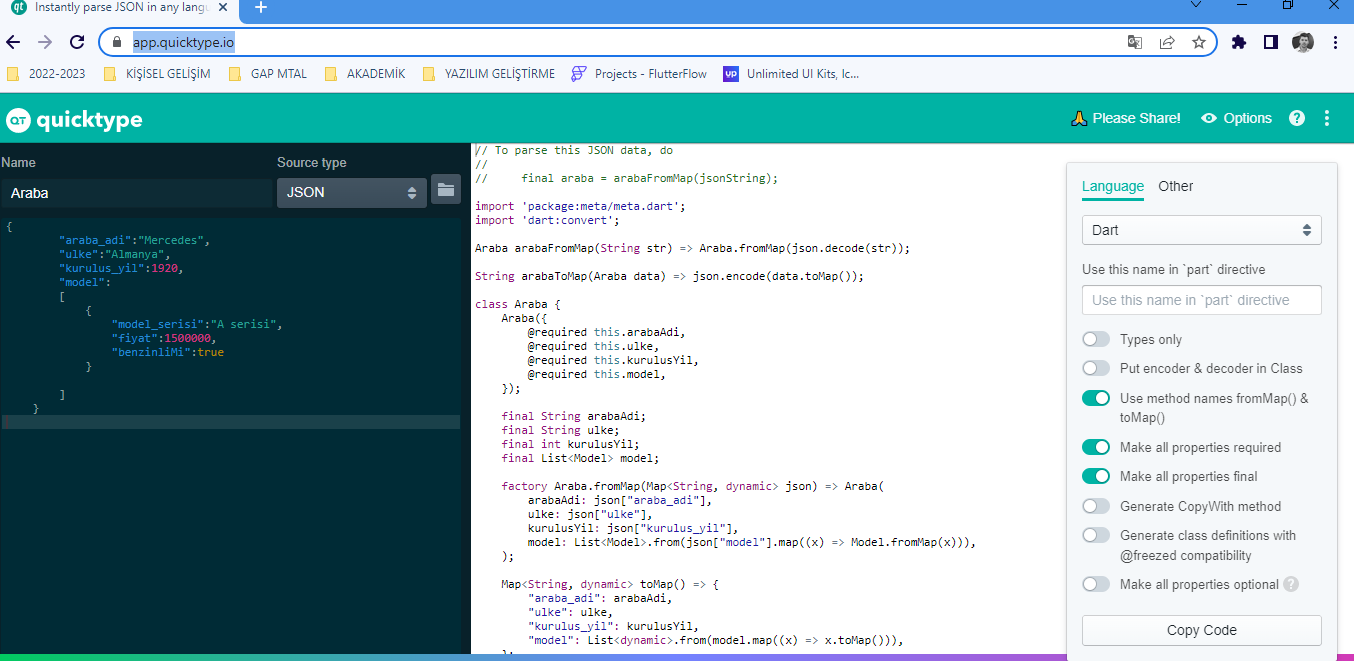
                "benzinliMi":true

            }

        ]

    }

11.QuickType.io ( <https://app.quicktype.io/> )adlı adreste model dönüşümü yapalım.kopyaladığımız kodu sol taraftaki kısma yapıştırıp model ismini Araba yapalım. Sağ taraftan da ilgili alanları aktif edelim. Copy Code diyerek oluşan model kodlarını araba\_model.dart a ekleyelim.



12.araba\_model.dart içeriği;

import 'dart:convert';

Araba arabaFromMap(String str) => Araba.fromMap(json.decode(str));

String arabaToMap(Araba data) => json.encode(data.toMap());

class Araba {

  Araba({

    required this.arabaAdi,

    required this.ulke,

    required this.kurulusYil,

    required this.model,

  });

  final String arabaAdi;

  final String ulke;

  final int kurulusYil;

  final List<Model> model;

  factory Araba.fromMap(Map<String, dynamic> json) => Araba(

        arabaAdi: json["araba\_adi"],

        ulke: json["ulke"],

        kurulusYil: json["kurulus\_yil"],

        model: List<Model>.from(json["model"].map((x) => Model.fromMap(x))),

      );

  Map<String, dynamic> toMap() => {

        "araba\_adi": arabaAdi,

        "ulke": ulke,

        "kurulus\_yil": kurulusYil,

        "model": List<dynamic>.from(model.map((x) => x.toMap())),

      };

}

class Model {

  Model({

    required this.modelSerisi,

    required this.fiyat,

    required this.benzinliMi,

  });

  final String modelSerisi;

  final int fiyat;

  final bool benzinliMi;

  factory Model.fromMap(Map<String, dynamic> json) => Model(

        modelSerisi: json["model\_serisi"],

        fiyat: json["fiyat"],

        benzinliMi: json["benzinliMi"],

      );

  Map<String, dynamic> toMap() => {

        "model\_serisi": modelSerisi,

        "fiyat": fiyat,

        "benzinliMi": benzinliMi,

      };

}

13.Şimdi local\_json.dart içerisinde arabalarOku içerisine aşağıdaki dönüşümü sağlayan kodları ekleyelim.

arabalarJsonOku() async {

    String okunanString = await DefaultAssetBundle.of(context)

        .loadString("assets/data/arabalar.json");

    var jsonArray = jsonDecode(okunanString);

    List arabaListesi = jsonArray;

    //debugPrint(arabaListesi[1]["model"][1].toString());

    List<Araba> tumArabalar =

        (jsonArray as List).map((arabaMap) => Araba.fromMap(arabaMap)).toList();

    debugPrint(tumArabalar.length.toString());

    //debugPrint(okunanString);

  }

14.Verileri ListView de listelemek için metodumuzu aşağıdaki gibi değiştirelim.

Future<List<Araba>> arabalarJsonOku() async {

    String okunanString = await DefaultAssetBundle.of(context)

        .loadString("assets/data/arabalar.json");

    var jsonArray = jsonDecode(okunanString);

    //List arabaListesi = jsonArray;

    //debugPrint(arabaListesi[1]["model"][1].toString());

    //debugPrint(okunanString);

    List<Araba> tumArabalar =

        (jsonArray as List).map((arabaMap) => Araba.fromMap(arabaMap)).toList();

    debugPrint(tumArabalar.length.toString());

    return tumArabalar;

  }

15.Uzun sürecek veri çekme işlemlerini tasarım ekranında görmek için FutureBuilder kullanmak gerekiyor.

local\_json.dart içerisinde body kısmını değiştirelim.

 body: FutureBuilder<List<Araba>>(

        future: arabalarJsonOku(),

        builder: (context, snapshot) {

          if (snapshot.hasData) {

            List<Araba> arabaListesi = snapshot.data!;

            return ListView.builder(

                itemCount: arabaListesi.length,

                itemBuilder: (context, index) {

                  return ListTile(

                      title: Text(arabaListesi[index].arabaAdi),

                      subtitle: Text(arabaListesi[index].ulke),

                      leading: CircleAvatar(

                        child:

                            Text(arabaListesi[index].model[0].fiyat.toString()),

                      ));

                });

          } else if (snapshot.hasError) {

            return Center(

              child: Text(snapshot.error.toString()),

            );

          } else {

            return const Center(

              child: CircularProgressIndicator(),

            );

          }

        },

      ),

16.Hata olması verilerin gelmemesi durumunda uygulama patlamasın diye try catch yapısı kullanalım.

Future<List<Araba>> arabalarJsonOku() async {

    try {

      /\* await Future.delayed(const Duration(seconds: 5), () {

        return Future.error('5 saniye sonra hata cıktı');

      });\*/

      debugPrint('5 saniyelik işlem başlıyor');

      await Future.delayed(const Duration(seconds: 5), () {

        debugPrint('5 saniyelik işlem bitti');

      });

      String okunanString = await DefaultAssetBundle.of(context)

          .loadString('assets/data/arabalar.json');

      var jsonArray = jsonDecode(okunanString);

      //debugPrint(okunanString);

      //debugPrint("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

      /\*

    List arabaListesi = jsonObject;

    debugPrint(arabaListesi[1]['model'][0]['fiyat'].toString());\*/

      List<Araba> tumArabalar = (jsonArray as List)

          .map((arabaMap) => Araba.fromMap(arabaMap))

          .toList();

      debugPrint(tumArabalar.length.toString());

      return tumArabalar;

    } catch (e) {

      debugPrint(e.toString());

      return Future.error(e.toString());

    }

  }

Not : Verileri arttırmak için arabalar.json içindeki verileri kopyala yapıştır ile çoğaltalım.

17. Verileri sürekli build ile çağırmamak için aşağıdaki metodu kullanalım.

class \_LocalJsonState extends State<LocalJson> {

  late final Future<List<Araba>> \_listeyiDoldur;

  @override

  void initState() {

    super.initState();

    \_listeyiDoldur = arabalarJsonOku();

  }

//body kısmına ekleyelim.

body: FutureBuilder<List<Araba>>(

            future: \_listeyiDoldur,

18.Veriler gelmeden boş ekran yerine belirli verileri görmek için initialdata yapısı kullanılır.

body: FutureBuilder<List<Araba>>(

            future: \_listeyiDoldur,

            initialData: [

              Araba(arabaAdi: "Egea", ulke: "İtalya", kurulusYil: 2018, model: [

                Model(modelSerisi: "Fire", fiyat: 200, benzinliMi: true)

              ]),

              Araba(

                  arabaAdi: "Toyota",

                  ulke: "Japonya",

                  kurulusYil: 2020,

                  model: [

                    Model(modelSerisi: "Corolla", fiyat: 400, benzinliMi: false)

                  ])

            ],

İnternetten Veri Çekme ( API )

Local json dosyasından veri çekme işlemleri ile hemen hemen aynıdır.Farkları;

1. Dart Add Dependency ile dio isimli paket indirilir.

pubspec.yaml içinde görülür

dio: ^4.0.6

1. main.dart adresine bir buton daha eklenir.

ElevatedButton(

              style: ElevatedButton.styleFrom(primary: Colors.orange),

              onPressed: () {

                Navigator.of(context).push(

                    MaterialPageRoute(builder: (context) => const RemoteApi()));

              },

              child: const Text('Remote Api')),

<https://jsonplaceholder.typicode.com/users> adresine gidilir. Ordaki json a ait bir kayıt model çıkarmak için <https://app.quicktype.io/> adresinden model oluşturulur.user\_model.dart a model atılır.

user\_model.dart içeriği;

// To parse this JSON data, do

//

//     final userModel = userModelFromMap(jsonString);

import 'dart:convert';

UserModel userModelFromMap(String str) => UserModel.fromMap(json.decode(str));

String userModelToMap(UserModel data) => json.encode(data.toMap());

class UserModel {

    UserModel({

        required this.id,

        required this.name,

        required this.username,

        required this.email,

        required this.address,

        required this.phone,

        required this.website,

        required this.company,

    });

    final int id;

    final String name;

    final String username;

    final String email;

    final Address address;

    final String phone;

    final String website;

    final Company company;

    factory UserModel.fromMap(Map<String, dynamic> json) => UserModel(

        id: json["id"],

        name: json["name"],

        username: json["username"],

        email: json["email"],

        address: Address.fromMap(json["address"]),

        phone: json["phone"],

        website: json["website"],

        company: Company.fromMap(json["company"]),

    );

    Map<String, dynamic> toMap() => {

        "id": id,

        "name": name,

        "username": username,

        "email": email,

        "address": address.toMap(),

        "phone": phone,

        "website": website,

        "company": company.toMap(),

    };

}

class Address {

    Address({

        required this.street,

        required this.suite,

        required this.city,

        required this.zipcode,

        required this.geo,

    });

    final String street;

    final String suite;

    final String city;

    final String zipcode;

    final Geo geo;

    factory Address.fromMap(Map<String, dynamic> json) => Address(

        street: json["street"],

        suite: json["suite"],

        city: json["city"],

        zipcode: json["zipcode"],

        geo: Geo.fromMap(json["geo"]),

    );

    Map<String, dynamic> toMap() => {

        "street": street,

        "suite": suite,

        "city": city,

        "zipcode": zipcode,

        "geo": geo.toMap(),

    };

  @override

  String toString() {

    return 'Address(street: $street, suite: $suite, city: $city, zipcode: $zipcode, geo: $geo)';

  }

}

class Geo {

    Geo({

        required this.lat,

        required this.lng,

    });

    final String lat;

    final String lng;

    factory Geo.fromMap(Map<String, dynamic> json) => Geo(

        lat: json["lat"],

        lng: json["lng"],

    );

    Map<String, dynamic> toMap() => {

        "lat": lat,

        "lng": lng,

    };

}

class Company {

    Company({

        required this.name,

        required this.catchPhrase,

        required this.bs,

    });

    final String name;

    final String catchPhrase;

    final String bs;

    factory Company.fromMap(Map<String, dynamic> json) => Company(

        name: json["name"],

        catchPhrase: json["catchPhrase"],

        bs: json["bs"],

    );

    Map<String, dynamic> toMap() => {

        "name": name,

        "catchPhrase": catchPhrase,

        "bs": bs,

    };

}

remote\_api.dart adında dosya oluşturulur.

import 'package:dio/dio.dart';

import 'package:flutter/material.dart';

import 'model/user\_model.dart';

class RemoteApi extends StatefulWidget {

  const RemoteApi({Key? key}) : super(key: key);

  @override

  \_RemoteApiState createState() => \_RemoteApiState();

}

class \_RemoteApiState extends State<RemoteApi> {

  Future<List<UserModel>> \_getUserList() async {

    try {

      var response =

          await Dio().get('https://jsonplaceholder.typicode.com/users');

      List<UserModel> \_userList = [];

      if (response.statusCode == 200) {

        \_userList =

            (response.data as List).map((e) => UserModel.fromMap(e)).toList();

      }

      return \_userList;

    } on DioError catch (e) {

      return Future.error(e.message);

    }

  }

  late final Future<List<UserModel>> \_userList;

  @override

  void initState() {

    super.initState();

    \_userList = \_getUserList();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text('Remote Api with Dio'),

      ),

      body: Center(

        child: FutureBuilder<List<UserModel>>(

          future: \_userList,

          builder: (context, snapshot) {

            if (snapshot.hasData) {

              var userList = snapshot.data!;

              return ListView.builder(

                itemBuilder: (context, index) {

                  var user = userList[index];

                  return ListTile(

                    title: Text(user.email),

                    subtitle: Text(user.address.toString()),

                    leading: Text(user.id.toString()),

                  );

                },

                itemCount: userList.length,

              );

            } else if (snapshot.hasError) {

              return Text(snapshot.error.toString());

            } else {

              return const CircularProgressIndicator();

            }

          },

        ),

      ),

    );

  }

}