**REST API**

Flutter cung cấp package http để sử dụng nguồn HTTP. http là một thư viện Future-based sử dụng tính năng await và async. Nó cung cấp phương thức cấp cao và đơn giản để phát triển REST trên ứng dụng di động.

Nội dung cơ bản:

Gói http cung cấp các lớp cấp cao và http request từ web

- Lớp http cung cấp chức năng để làm việc với tất cả các kiểu dữ liệu HTTP được request

- Phương thức http có sử dụng url , và bổ sung thông tin thông qua Dart Map ( post dữ liệu, bổ sung tiêu đề, ...). Nó yêu cầu lên máy chủ và thu thập phản hồi với async/await. Ví dụ đoạn code dưới đây đọc dữ liệu từ url và in nó trên console

print(await http.read('https://flutter.dev/'));

Một vài phương thức chính :

- read : gởi yêu cầu lên sever thông qua phương thức GET và trả về  Future<String>

- get : gởi yêu cầu lên sever thông qua phương thức GET và trả về Future<Response>. Response là lớp giữ lại các thông tin phản hồi

- post : gởi yêu cầu lên sever thông qua phương thức POST  bằng việc đưa giá trị lên sever và phản hồi Future<Response>

- put : gởi yêu cầu lên sever thông qua phương thức PUT và trả về phản hồi như Future<Response>

- head : gởi yêu cầu lên sever thông qua phương thức HEAD và trả về phản hồi như Future<Response>

- delete : gởi yêu cầu lên sever thông qua phương thức DELETE và trả về phản hồi như Future<Response>

Http cũng cung cấp nhiều lớp standard HTTP client. client có nhiệm vụ hỗ trợ kết nối. Nó sẽ hữu ích khi có rất nhiều request lên sever

var client = new http.Client();

try {

print(await client.get('https://flutter.dev/'));

}

finally {

client.close();

}

Truy cập vào Product service API

Ta sẽ tạo ứng dụng đơn giản để lấy dữ liệu Product từ web server và sau đó hiển thị danh sách Product trong ListView

Tạo ứng dụng Flutter trong android với tên "*product\_rest\_app"*

Copy assets dưới đây vào pubspec.yaml ,(các bạn có thể lấy ảnh ở bài 7 )

flutter:

assets:

- assets/appimages/floppy.jpg

- assets/appimages/iphone.jpg

- assets/appimages/laptop.jpg

- assets/appimages/pendrive.jpg

- assets/appimages/pixel.jpg

- assets/appimages/tablet.jpg

Thêm http package vào file pubspec.yaml như sau :

dependencies:

http: ^0.12.0+2

Thêm một vào package trong hàm main.dart

import 'dart:async';

import 'dart:convert';

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

Tạo file JSON, product.json

[

{

"name": "iPhone",

"description": "iPhone is the stylist phone ever",

"price": 1000,

"image": "iphone.png"

},

{

"name": "Pixel",

"description": "Pixel is the most feature phone ever",

"price": 800,

"image": "pixel.png"

},

{

"name": "Laptop",

"description": "Laptop is most productive development tool",

"price": 2000,

"image": "laptop.png"

},

{

"name": "Tablet",

"description": "Tablet is the most useful device ever for meeting",

"price": 1500,

"image": "tablet.png"

},

{

"name": "Pendrive",

"description": "Pendrive is useful storage medium",

"price": 100,

"image": "pendrive.png"

},

{

"name": "Floppy Drive",

"description": "Floppy drive is useful rescue storage medium",

"price": 20,

"image": "floppy.png"

}

]

Tạo thư mục mới tên là JSONWebServer , đặt products.json vào trong

Chạy bất kì web server với JSONWebServer và lấy đường dẫn. Ví dụ http://192.168.184.1:8000/products.json. Một vài web server như apache, nginx, ...

Cách đơn giản nhất là cài đặt nodejs dựa trên ứng dụng http-server. Các bước cài đặt như sau :

- Cài đặt ứng dụng NodeJs https://nodejs.org/en

- Đi đến thư mục JSONWebServer

cd /path/to/JSONWebServer

- Cài đặt http-server package bằng cách sử dụng npm

npm install -g http-server

- Sau đó thử chạy server

http-server . -p 8000

Starting up http-server, serving .

Available on:

http://192.168.99.1:8000

http://127.0.0.1:8000

Hit CTRL-C to stop the server

Tạo file mới Product.dart trong lớp Product

Viết factory constructor trong lớp Product , Product.fromMap dùng để chuyển đổi dữ liệu map trong đối tượng Product . Thông thường, tệp JSON sẽ được chuyển đổi bên trong đối tượng Dart Map và sau đó chuyển đổi sang đối tượng liên qua (Product)

factory Product.fromJson(Map<String, dynamic> data) {

return Product(

data['name'],

data['description'],

data['price'],

data['image'],

);

}

Code trong hàm product.dart như sau :

class Product {

final String name;

final String description;

final int price;

final String image;

Product(this.name, this.description, this.price, this.image);

factory Product.fromMap(Map<String, dynamic> json) {

return Product(

json['name'],

json['description'],

json['price'],

json['image'],

);

}

}

Bây giờ ta sẽ viết 2 phương thức - parseProducts và fetchProducts - trong lớp chính để lấy và tải thông tin sản phẩm từ web server(máy chủ) trong List<Product>

List<Product> parseProducts(String responseBody) {

final parsed = json.decode(responseBody).cast<Map<String, dynamic>>();

return parsed.map<Product>((json) =>Product.fromJson(json)).toList();

}

Future<List<Product>> fetchProducts() async {

final response = await http.get('http://192.168.1.2:8000/products.json');

if (response.statusCode == 200) {

return parseProducts(response.body);

} else {

throw Exception('Unable to fetch products from the REST API');

}

}

Ta nên lưu ý một vài điểm sau :

-Future được sử dụng để trì hoãn việc tải thông tin sản phẩm cho đến khi cần thiết

- http.get được dùng để lấy dữ liệu từ internet

- json.decode được sử dụng để dịch dữ liệu JSON trong Dart Map. Mỗi một dữ liệu JSON được dịch , nó sẽ chuyển vào List<Product> bằng fromMap của lớp Product

Trong lớp MyApp, thêm giá trị product mới, các product thuộc kiểu Future<Product> và đưa vào hàm constructor

class MyApp extends StatelessWidget {

final Future<List<Product>> products;

MyApp({Key key, this.products}) : super(key: key);

...

Trong lớp MyHomePage, thêm một vài Product thuộc kiểu Future<Product> và đưa vào constructor .

class MyHomePage extends StatelessWidget {

final String title;

final Future<ListList<Product>> products;

MyHomePage({Key key, this.title, this.products}) : super(key: key);

...

 Trong widget home (MyHomePage)  thay đổi như sau

home: MyHomePage(title: 'Product Navigation demo home page', products: products),

Thay đổi hàm chính trên StateFullWidge

void main() => runApp(MyApp(fetchProduct()));

Tạo widget mới là ProductBoxList để xây dựng list Product trong home page

class ProductBoxList extends StatelessWidget {

final List<Product> items;

ProductBoxList({Key key, this.items});

@override

Widget build(BuildContext context) {

return ListView.builder(

itemCount: items.length,

itemBuilder: (context, index) {

return GestureDetector(

child: ProductBox(item: items[index]),

onTap: () {

Navigator.push(

context, MaterialPageRoute(

builder: (context) =gt; ProductPage(item: items[index]),

),

);

},

);

},

);

}

}

Lưu ý rằng,  chúng ta sử dụng cùng nội dung trong ứng dụng Navigation để đưa ra list Product

Cuối cùng ta sửa đổi *MyHomePage*widget's để lấy thông tin Product sử dụng tính năng Future thay vì phương thức gọi thông thường

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Product Navigation")),

body: Center(

child: FutureBuilder<List<Product>>(

future: products, builder: (context, snapshot) {

if (snapshot.hasError) print(snapshot.error);

return snapshot.hasData ? ProductBoxList(items: snapshot.data)

// return the ListView widget :

Center(child: CircularProgressIndicator());

},

),

)

);

}

- Ở đây ta đã sử dụng widget FutureBuilder để render widget . FutureBuilder sẽ cố lấy dữ liệu từ thuộc tính future (thuộc Future<List<Product>>) . Nếu thuộc tính future trả dữ liệu về , nó sẽ render widget sử dụng ProductBoxList , mặt khác sẽ  ném lỗi(throw err)

Toàn bộ code trong hàm main.dart như dưới đây :

import 'package:flutter/material.dart';

import 'dart:async';

import 'dart:convert';

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

import 'Product.dart';

void main() => runApp(MyApp(products: fetchProducts()));

List<Product> parseProducts(String responseBody) {

final parsed = json.decode(responseBody).cast<Map<String, dynamic>>();

return parsed.map<Product>((json) => Product.fromMap(json)).toList();

}

Future<List<Product>> fetchProducts() async {

final response = await http.get('http://192.168.1.2:8000/products.json');

if (response.statusCode == 200) {

return parseProducts(response.body);

} else {

throw Exception('Unable to fetch products from the REST API');

}

}

class MyApp extends StatelessWidget {

final Future<List<Product>> products;

MyApp({Key key, this.products}) : super(key: key);

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Demo',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyHomePage(title: 'Product Navigation demo home page', products: products),

);

}

}

class MyHomePage extends StatelessWidget {

final String title;

final Future<List<Product>> products;

MyHomePage({Key key, this.title, this.products}) : super(key: key);

// final items = Product.getProducts();

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text("Product Navigation")),

body: Center(

child: FutureBuilder<List<Product>>(

future: products, builder: (context, snapshot) {

if (snapshot.hasError) print(snapshot.error);

return snapshot.hasData ? ProductBoxList(items: snapshot.data)

// return the ListView widget :

Center(child: CircularProgressIndicator());

},

),

)

);

}

}

class ProductBoxList extends StatelessWidget {

final List<Product> items;

ProductBoxList({Key key, this.items});

@override

Widget build(BuildContext context) {

return ListView.builder(

itemCount: items.length,

itemBuilder: (context, index) {

return GestureDetector(

child: ProductBox(item: items[index]),

onTap: () {

Navigator.push(

context, MaterialPageRoute(

builder: (context) => ProductPage(item: items[index]),

),

);

},

);

},

);

}

}

class ProductPage extends StatelessWidget {

ProductPage({Key key, this.item}) : super(key: key);

final Product item;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text(this.item.name),),

body: Center(

child: Container(

padding: EdgeInsets.all(0),

child: Column(

mainAxisAlignment: MainAxisAlignment.start,

crossAxisAlignment: CrossAxisAlignment.start,

children: <Widget>[

Image.asset("assets/appimages/" + this.item.image),

Expanded(

child: Container(

padding: EdgeInsets.all(5),

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Text(this.item.name, style:

TextStyle(fontWeight: FontWeight.bold)),

Text(this.item.description),

Text("Price: " + this.item.price.toString()),

RatingBox(),

],

)

)

)

]

),

),

),

);

}

}

class RatingBox extends StatefulWidget {

@override

\_RatingBoxState createState() =>\_RatingBoxState();

}

class \_RatingBoxState extends State<RatingBox> {

int \_rating = 0;

void \_setRatingAsOne() {

setState(() {

\_rating = 1;

});

}

void \_setRatingAsTwo() {

setState(() {

\_rating = 2;

});

}

void \_setRatingAsThree() {

setState(() {

\_rating = 3;

});

}

Widget build(BuildContext context) {

double \_size = 20;

print(\_rating);

return Row(

mainAxisAlignment: MainAxisAlignment.end,

crossAxisAlignment: CrossAxisAlignment.end,

mainAxisSize: MainAxisSize.max,

children: <Widget>[

Container(

padding: EdgeInsets.all(0),

child: IconButton(

icon: (

\_rating >= 1

? Icon(Icons.star, ize: \_size,)

: Icon(Icons.star\_border, size: \_size,)

),

color: Colors.red[500], onPressed: \_setRatingAsOne, iconSize: \_size,

),

),

Container(

padding: EdgeInsets.all(0),

child: IconButton(

icon: (

\_rating >= 2

? Icon(Icons.star, size: \_size,)

: Icon(Icons.star\_border, size: \_size, )

),

color: Colors.red[500],

onPressed: \_setRatingAsTwo,

iconSize: \_size,

),

),

Container(

padding: EdgeInsets.all(0),

child: IconButton(

icon: (

\_rating >= 3 ?

Icon(Icons.star, size: \_size,)

: Icon(Icons.star\_border, size: \_size,)

),

color: Colors.red[500],

onPressed: \_setRatingAsThree,

iconSize: \_size,

),

),

],

);

}

}

class ProductBox extends StatelessWidget {

ProductBox({Key key, this.item}) : super(key: key);

final Product item;

Widget build(BuildContext context) {

return Container(

padding: EdgeInsets.all(2), height: 140,

child: Card(

child: Row(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Image.asset("assets/appimages/" + this.item.image),

Expanded(

child: Container(

padding: EdgeInsets.all(5),

child: Column(

mainAxisAlignment: MainAxisAlignment.spaceEvenly,

children: <Widget>[

Text(this.item.name, style:TextStyle(fontWeight: FontWeight.bold)),

Text(this.item.description),

Text("Price: " + this.item.price.toString()),

RatingBox(),

],

)

)

)

]

),

)

);

}

}

Okeyy, chạy ứng dụng và xem kết quả nhé. Nó sẽ cho ra cùng kết quả như ứng dụng