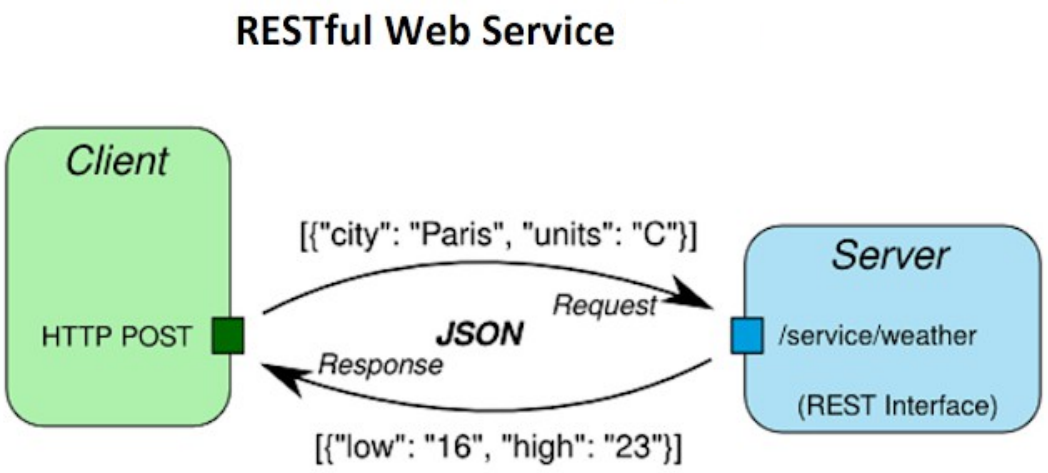
Flutter working with *REST APIs*

By Prasertsak U., Ph.D

What is REST API ?

- REST API (also known as RESTful API) is an application programming interface (API or web API).
  - A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
  - Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.



Picture from: https://www.java67.com/

What is REST API ? *(cont.)*

- REST (Representational State Transfer) เป็นสไตล์การออกแบบสถาปัตยกรรมสำหรับสร้าง Web Services
- REST API คือช่องทางหรืออินเทอร์เฟซที่แอปสามารถสื่อสารกับ Server เพื่อดึงข้อมูลหรือส่งข้อมูลไปยังเซิร์ฟเวอร์ได้
- REST API ใช้ HTTP Methods เป็นหลัก ได้แก่
  - GET: ดึงข้อมูล
  - POST: ส่งข้อมูลใหม่
  - PUT: อัปเดตข้อมูล
  - DELETE: ลบข้อมูล
- การตอบสนองมีสถานะ เช่น 200 (สำเร็จ), 404 (ไม่พบข้อมูล), 400 (คำขอผิดพลาด) และอื่นๆ
- ข้อมูลมักถูกส่งในรูปแบบ JSON ซึ่งง่ายต่อการใช้งาน

JSON

- **JSON** : JavaScript Object Notation คือรูปแบบการเก็บข้อมูลแบบข้อความในลักษณะของ ***Key : Value*** โดยใช้สัญลักษณ์เชิงอ็อปเจ็คมาประกอบ
  เพื่อให้สามารถนำไปใช้กับภาษาคอมพิวเตอร์ต่าง ๆ ได้อย่างสะดวก เช่น

```
{ "name"   : "John Smith",
  "sku"    : "20223",
  "price"  : 23.95,
  "shipTo" : { "name" : "Jane Smith",
               "address" : "123 Maple Street",
               "city" : "Pretendville",
               "state" : "NY",
               "zip"   : "12345" },
  "billTo" : { "name" : "John Smith",
               "address" : "123 Maple Street",
               "city" : "Pretendville",
               "state" : "NY",
               "zip"   : "12345" }
}
```

```
{"users": [
  {"username" : "SammyShark", "location" : "Indian Ocean"},
  {"username" : "JesseOctopus", "location" : "Pacific Ocean"},
  {"username" : "DrewSquid", "location" : "Atlantic Ocean"},
  {"username" : "JamieMantisShrimp", "location" : "Pacific Ocean"}
] }
```

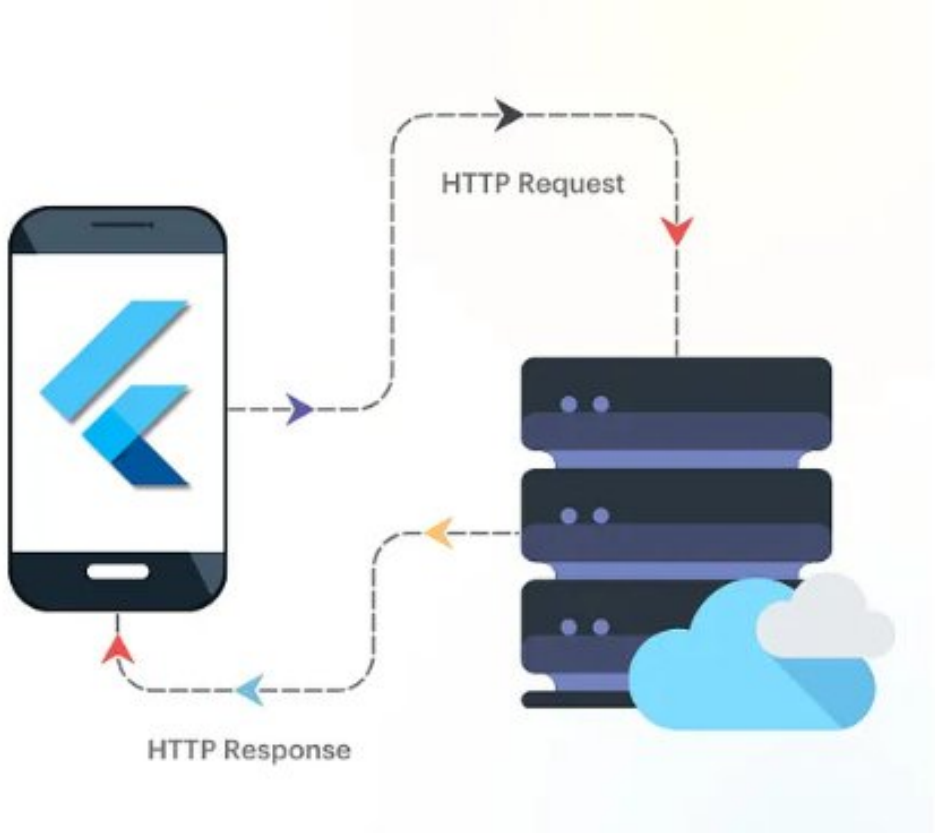
ขั้นตอนการเรียกใช้ REST API ใน Flutter

- เพิ่ม plugin ชื่อ ***http*** ในส่วน dependencies ภายในไฟล์ pubspec.yaml เพื่อใช้ในการส่งคำขอผ่านและดึงข้อมูลผ่าน HTTP
  *(package ชื่อ **dio** เป็นอีก package ที่สามารถใช้ได้ แต่มีความซับซ้อนมากกว่าเหมาะกับการเรียก API ที่มีความละเอียดและซับซ้อน)*
- ใช้ async/await เพื่อการจัดการคำขอแบบ asynchronous โดยไม่บล็อก UI
- แปลงข้อมูล JSON หรือการ decode JSON ที่ได้รับเป็น Dart object (Model) เพื่อการใช้งานที่ง่ายขึ้น (เป็นทางเลือกที่ทำให้ง่ายขึ้น ไม่ได้บังคับ)
- ใช้ try-catch เพื่อจัดการกับข้อผิดพลาดที่อาจจะเกิดขึ้น
- ควรแยก logic ในส่วนที่เรียกใช้ API ออกจากส่วนที่เป็น UI (เช่น ทำเป็น Service Class)

Setting Up the HTTP Package

- need to install the ***http*** package, which simplifies sending and receiving HTTP request/response in Flutter.
- Open your *pubspec.yaml* file and add the http package under dependencies:

```
dependencies:
  http: ^1.4.0
```



API ที่ใช้ในการทดสอบและเรียนรู้

- Using http://ip-api.com/
- Sample endpoint http://ip-api.com/json/158.108.0.0

API นี้ไม่รองรับ **https**

IP Geolocation API

Fast, accurate, reliable

Free for non-commercial use, no API key required

Easy to integrate, available in JSON, XML, CSV, Newline, PHP

Serving more than 1 billion requests per day, trusted by thousands of businesses

Structure of JSON message

ตัวอย่างโครงสร้าง JSON ที่มีการรับส่งกับ API นี้

☑	status	success or fail


```
{
  "status": "success",
  "country": "Thailand",
  "countryCode": "TH",
  "region": "10",
  "regionName": "Bangkok",
  "city": "Chatuchak",
  "zip": "10800",
  "lat": 13.8509,
  "lon": 100.557,
  "timezone": "Asia/Bangkok",
  "isp": "Kasetsart University",
  "org": "Kasetsart University",
  "as": "AS9411 Kasetsart University, Thailand",
  "query": "158.108.0.0"
}
```


An example of Fail


```
1  {
2      "status": "fail",
3      "message": "SSL unavailable for this endpoint, order a key at https://members.ip-api.com/"
4  }
```


Quick test

You can edit this query and experiment with the options

GET http://ip-api.com/json/158.108.0.0

response


```
{
  "query": "158.108.0.0",
  "status": "success",
  "country": "Thailand",
  "countryCode": "TH",
  "region": "10",
  "regionName": "Bangkok",
  "city": "Chatuchak",
  "zip": "10800",
  "lat": 13.8509,
  "lon": 100.557,
  "timezone": "Asia/Bangkok",
  "isp": "Kasetsart University",
  "org": "Kasetsart University",
  "as": "AS9411 Kasetsart University, Thailand"
}
```

สร้าง Class ที่ใช้เป็น Model เพื่อทำงานกับ JSON

- สร้าง Model Class เพื่อทำงานกับ JSON Message ที่ส่งกลับมาจาก API โดยใช้การแปลงข้อมูล JSON ไปเป็น Class ภาษา Dart โดยอัตโนมัติ
- ขั้นตอนที่ 1: นำโครงสร้างตัวอย่าง JSON มาใช้เป็นตัวเริ่มต้น ด้วยการทดสอบผลลัพธ์จากการเรียกใช้ API เป้าหมาย เช่น http://ip-api.com/json/158.108.0.0
- ขั้นตอนที่ 2: เข้าเว็ปไซต์ https://javiercbk.github.io/json_to_dart/ จากนั้นให้วาง Response Message ที่เป็น JSON ลงในช่อง และทำการ Generate เป็น Class ในภาษา Dart

จะต้องทำการทดสอบเรียกใช้ API ก่อนเพื่อให้ได้ข้อความตอบกลับที่เป็น JSON จากนั้นจึงนำ JSON Message ที่ได้ไปใช้ในการสร้าง Class เพื่อรองรับอีกครั้ง

ตัวอย่างการสร้าง Model Class

- ตัวอย่าง class ที่ได้จากการสร้างโดยอัตโนมัติจากข้อมูล JSON
- สามารถเลือก copy to clipboard เพื่อนำมาใส่ใน Flutter project ได้

การใช้เครื่องมือในการสร้างโค้ดเป็นตัวช่วยอำนวยความสะดวกในการจัดการกับข้อมูล JSON ให้รวดเร็วขึ้นเท่านั้น
ไม่ใช่แนวทางเดียว เราสามารถใช้แนวทางอื่นในการจัดการได้อีก

JSON to Dart

Paste your JSON in the textarea below, click convert and get your Dart classes for free.

JSON

```
{
    "status": "success",
    "country": "Thailand",
    "countryCode": "TH",
    "region": "10",
    "regionName": "Bangkok",
    "city": "Chatuchak",
    "zip": "10800",
    "lat": 13.8509,
    "lon": 100.557,
    "timezone": "Asia/Bangkok",
    "isp": "Kasetsart University",
    "org": "Kasetsart University",
    "as": "AS9411 Kasetsart University, Thaila
    "query": "158.108.0.0"
}
```

Geolocation

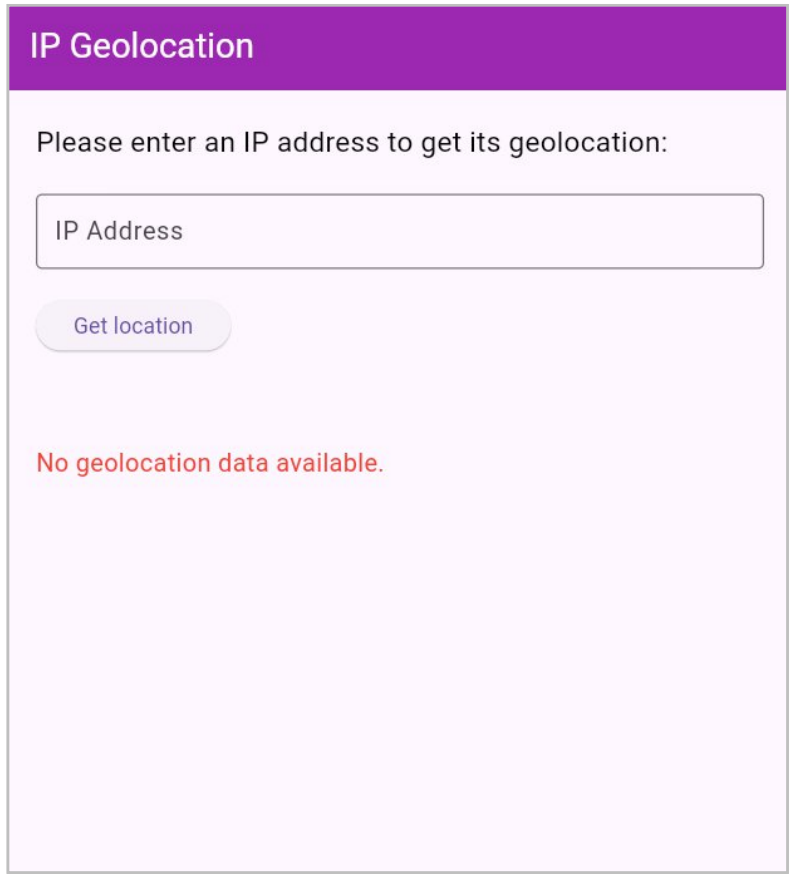
Generate Dart ☐ Use private fields

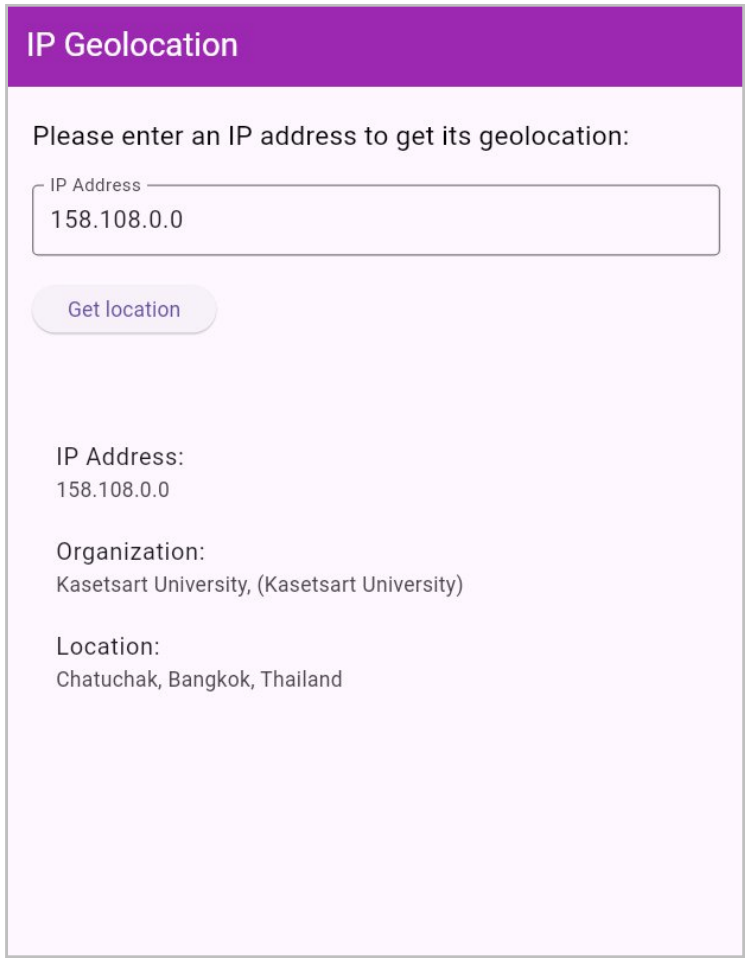
Copy Dart code to clipboard

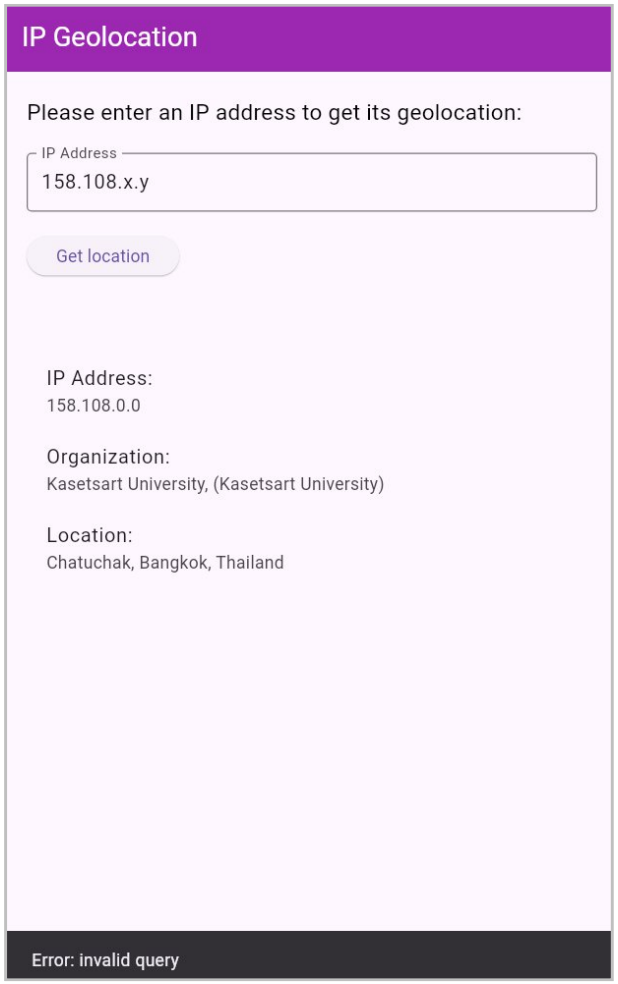
```
class Geolocation {
  String? status;
  String? country;
  String? countryCode;
  String? region;
  String? regionName;
  String? city;
  String? zip;
  double? lat;
  double? lon;
  String? timezone;
  String? isp;
  String? org;
  String? as;
  String? query;

  Geolocation(
      {this.status,
      this.country,
      this.countryCode,
      this.region,
      this.regionName,
      this.city,
      this.zip,
      this.lat,
```

ตัวอย่างภาพ Flutter App ที่ต้องการพัฒนา







ตัวอย่าง Flutter App (*#1*)

- สร้าง Project ใหม่ชื่อว่า *test_calling_api*
- **สร้าง folder ภายใน /lib เพิ่มเติม 3 folders ได้แก่ pages, models และ services**
- สร้างไฟล์ชื่อ *home.dart* ไว้ภายใน lib/pages
- สร้างไฟล์ชื่อ *geolocation_model.dart* ไว้ภายใน lib/models
- แก้ไข *main.dart* ให้เรียกใช้ home.dart เป็นหน้าเริ่มต้น ดังนี้

main.dart

```dart
import 'package:flutter/material.dart';
import 'pages/home.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      home: HomePage(),
    );
  }
}
```

ตัวอย่าง Flutter App (#2)

- นำเนื้อหาของคลาสที่สร้างจาก JSON Message หรือ Model ของ JSON จากเว็บไซต์มาใส่ลงในไฟล์ *models/geolocation_model.dart*

```dart
class Geolocation {
  String? status;
  String? country;
  String? countryCode;
  String? region;
  String? regionName;
  String? city;
  String? zip;
  double? lat;
  double? lon;
  String? timezone;
  String? isp;
  String? org;
  String? as;
  String? query;
  String? message;
```

```dart
  Geolocation({
    this.status,
    this.country,
    this.countryCode,
    this.region,
    this.regionName,
    this.city,
    this.zip,
    this.lat,
    this.lon,
    this.timezone,
    this.isp,
    this.org,
    this.as,
    this.query,
    this.message,
  });
```

```dart
  Geolocation.fromJson(Map<String, dynamic> json) {
    status = json['status'];
    country = json['country'] == null ? '' : json['country'].toString();
    countryCode = json['countryCode'] == null ? '' : json['countryCode'].toString();
    region = json['region'] == null ? '' : json['region'].toString();
    regionName = json['regionName'] == null ? '' : json['regionName'].toString();
    city = json['city'] == null ? '' : json['city'].toString();
    zip = json['zip'] == null ? '' : json['zip'].toString();
    lat = json['lat'] == null ? 0.0 : double.tryParse(json['lat'].toString()) ?? 0.0;
    lon = json['lon'] == null ? 0.0 : double.tryParse(json['lon'].toString()) ?? 0.0;
    timezone = json['timezone'] == null ? '' : json['timezone'].toString();
    isp = json['isp'] == null ? '' : json['isp'].toString();
    org = json['org'] == null ? '' : json['org'].toString();
    as = json['as'] == null ? '' : json['as'].toString();
    query = json['query'] == null ? '' : json['query'].toString();
    message = json['message'] == null ? '' : json['message'].toString();
  }
```

More...

ตัวอย่าง Flutter App (#3)

- สร้างไฟล์ *api_services.dart* ไว้ภายใน *lib/services*
- แก้ไข *api_services.dart* โดยการใส่ method ต่างๆ ที่จำเป็นต่อการเรียกใช้ API

api_services.dart

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

class Apiservices {
  // Add your API service methods here

  Future<Geolocation> requestGeolocation(String ipAddress) async {
    String apiUrl = 'http://ip-api.com/json/$ipAddress';
    try {
      return await http
        .get(Uri.parse(apiUrl))
        .then((value) {
          if (value.statusCode == 200) {
            // If the server returns an OK response, parse the JSON
            return processResponse(value.body);
          } else {
            // If the server did not return a 200 OK response, throw an exception
            throw Exception('Failed to load geolocation data');
          }
        })
        .catchError((error) {
          // Handle any errors that occur during the request
          throw Exception('Failed to request geolocation api');
        });
    } catch (e) {
      // Handle any exceptions that occur during the request
      throw Exception('Failed to get geolocation: $e');
    }
  }

  Geolocation processResponse(String responseBody) {
    return Geolocation.fromJson(jsonDecode(responseBody));
  }
}
```

async, await และ Future<T>

- async และ await จะใช้คู่กัน คือการจัดการกับ "เวลา"
  - async (Asynchronous): ใส่ไว้ที่ฟังก์ชันเพื่อบอกว่า "ฟังก์ชันนี้มีการทำงานที่ต้องรอ" และจะคืนค่ากลับไปเป็น Future เสมอ
  - await (Wait): ใส่ไว้หน้าคำสั่งที่ต้องใช้เวลารอ (เช่น การดึงข้อมูลจาก API) เพื่อบอกโปรแกรมว่า "ให้หยุดรอตรงนี้จนกว่าจะได้ข้อมูลกลับมาหรือทำงานเสร็จ ค่อยไปบรรทัดถัดไป"
- Future<T> คือ "สัญญา" ว่าเราจะได้ข้อมูลประเภท T ในอนาคต (***T*** คือชนิดของข้อมูลที่จะต้องระบุ เช่น double, int, bool หรือ object ของ Model Class ที่สร้างขึ้น และอื่น ๆ เป็นต้น)
- สำหรับผู้ที่เรียกใช้ฟังกชันที่มีการส่งค่ากลับเป็น Future<T> ก็มักจะต้องระบุ await ไว้ด้านหน้าด้วย เพื่อรอการส่งค่ากลับ เช่นกัน และเมื่อมีการใช้ await ที่ใด ที่ชื่อของฟังก์ชันนั้นก็จะต้องระบุ async ไว้ด้วยเสมอ

ตัวอย่าง Flutter App (#4)

- แก้ไข *home.dart* ให้แสดง UI เพื่อรับข้อมูลหมายเลข IP และแสดงปุ่มเพื่อ submit ข้อมูลไปให้กับ API ผ่าน service ที่สร้างไว้ก่อนหน้านี้

บรรทัดที่ 1-10

```
import 'package:flutter/material.dart';
import '../services/api_services.dart';
import '../models/geolocation_model.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}
```

บรรทัดที่ 11-35

```
class _HomePageState extends State<HomePage> {
  late Geolocation ipLocation;

  @override
  void initState() {
    super.initState();
    ipLocation = Geolocation(
      status: '',
      country: '',
      countryCode: '',
      region: '',
      regionName: '',
      city: '',
      zip: '',
      lat: 0.0,
      lon: 0.0,
      timezone: '',
      isp: '',
      org: '',
      as: '',
      query: '',
      message: '',
    );
  }
```

ตัวอย่าง Flutter App (#5)

- แก้ไข *home.dart* (ต่อ)

บรรทัดที่ 36-48

```
@override
  Widget build(BuildContext context) {
    String ipAddress = '';

    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'IP Geolocation',
          style: TextStyle(color: Colors.white),
        ),
        backgroundColor: Colors.purple,
      ),
```

บรรทัดที่ 49-73

```
body: Center(
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Padding(
            padding: const EdgeInsets.all(20.0),
            child: const Text(
              'Please enter an IP address to get its geolocation:',
              style: TextStyle(fontSize: 18, color: Colors.black),
            ),
          ),
          Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20.0),
            child: TextField(
              decoration: InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'IP Address',
                hintText: 'Enter a valid IP address',
              ),
              keyboardType: TextInputType.text,
              onChanged: (value) {
                ipAddress = value; // Capture the input IP address
              },
            ),
          ),
```

ตัวอย่าง Flutter App (#6)

- แก้ไข *home.dart (ต่อ)*

บรรทัดที่ 74-90

```
Padding(
        padding: const EdgeInsets.all(20.0),
        child: ElevatedButton(
          onPressed: () async {
            // Action when button is pressed
            await Apiservices()
                .requestGeolocation(ipAddress)
                .then((Geolocation geolocation) {
                  manageResponse(geolocation);
                })
                .catchError((error) {
                  showSnackBar('Error: $error');
                });
          },
          child: const Text('Get location'),
        ),
      ),
```

บรรทัดที่ 91-116

```
SizedBox(height: 20),
        ipLocation.status == 'success'
          ? Container(
            padding: const EdgeInsets.all(20.0),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                ListTile(
                  title: const Text('IP Address:'),
                  subtitle: Text('${ipLocation.query}'),
                ),
                ListTile(
                  title: const Text('Organization:'),
                  subtitle: Text(
                    '${ipLocation.org}, (${ipLocation.isp})',
                  ),
                ),
                ListTile(
                  title: const Text('Location:'),
                  subtitle: Text(
                    '${ipLocation.city}, ${ipLocation.regionName}, ${ipLocation.country}',
                  ),
                ),
              ],
            ),
          )
```

ตัวอย่าง Flutter App (#7)

- แก้ไข *home.dart (ต่อ)*

บรรทัดที่ 117-128

```
: Padding(
            padding: const EdgeInsets.all(20.0),
            child: const Text(
              'No geolocation data available.',
              style: TextStyle(fontSize: 16, color: Colors.red),
            ),
          ),
      ],
    ),
  ),
);
}
```

บรรทีดที่ 129-End

```
void manageResponse(Geolocation geolocation) {
  setState(() {
    ipLocation = geolocation;
  });
  if (geolocation.status != 'success') {
    showSnackBar('Error: ${geolocation.message}');
  }
}

void showSnackBar(String message) {
  ScaffoldMessenger.of(
    context,
  ).showSnackBar(SnackBar(content: Text(message)));
}
} // end of class
```

ทดสอบความเข้าใจ

- ให้นิสิตสร้าง Project ใหม่ และเรียกใช้ API ตามที่ระบุต่อไปนี้
- ให้นิสิตทดลองเรียกใช้ API ตาม endpoint นี้ และแสดงผลลัพธ์ที่ได้

https://dog.ceo/api/breeds/image/random

- ตัวอย่างข้อมูล JSON Message ที่ได้รับ

```
{
  "message": "https://images.dog.ceo/breeds/sheepdog-shetland/n02105855_14653.jpg",
  "status": "success"
}
```

link ของไฟล์ภาพของสุนัขที่สุ่มได้

หลังจากเรียกใช้ API ให้นำผลลัพธ์ที่ได้มาประมวลผล
เพื่อแสดงภาพสุนัขที่สุ่มได้บนหน้าจอ

การบ้าน #4

* หากมีการใช้ Provider Package ช่วยในการทำ State Management จะดีมาก

- ให้สร้าง App เกี่ยวกับ Article Reader โดยส่วนหัว (Header): ให้แสดงชื่อบทความ ส่วนเนื้อหาให้ใส่เนื้อหาที่เป็น Paragraph ยาวๆ ที่จำลองขึ้นเอง เป็นเรื่องเกี่ยวกับ อะไรก็ได้ (ให้สามารถ Scroll ได้ด้วยหากความยาวเกินขนาดหน้าจอ) ทั้งนี้เพื่อให้ เห็นผลของการเปลี่ยนชื่อ Font ขนาดตัวอักษร (Size) และ Mode แสดงผลได้ชัดเจน
- ภายใน App จะต้องมีจุดสำหรับเปิดหน้าจอ Setting โดยผู้ใช้ต้องสามารถกำหนดสิ่งเหล่านี้ได้
  - Font Family: เปลี่ยนชื่อ Font ตามค่าที่เลือก
  - Font Size: ปรับขนาดตัวอักษรระหว่าง **Small** (14px) และ **Large** (20px)
  - Display Mode: เปลี่ยนสีตาม **Light Mode** / **Dark Mode**
- หลังจากตั้งค่าเรียบร้อยแล้ว เมื่อกลับออกมาที่หน้าหลัก ทั้ง App จะต้องเปลี่ยนแปลงทันที ตามที่ตั้งค่าไว้ โดยไม่ต้องออกจาก App

* เมื่อออกจาก App แล้วเข้าใหม่ค่าที่ตั้งไว้ล่าสุดจะยังไม่ถูกนำมาใช้ เนื่องจากยังไม่มีการบันทึกเก็บไว้ (รอ Next Chapter..!)

* ส่งภายในชั่วโมงเรียนครั้งถัดไป