

# MOBILE PROGRAMMING WITH FLUTTER - DAY 6

เริ่มพัฒนาแอปพลิเคชันบนมือถือด้วย Flutter

# การบ้านสำหรับเช็คชื่อรอบนี้

ตรวจรายละเอียดแผนการจัดทำ

PROJECT รวมกลุ่ม 3 คน

เรากำลังทำอะไรใน “API 101”



Design & theming
Interactivity
Assets & media
Navigation & routing
Animations & transitions
Accessibility
Internationalization
<b>Beyond UI</b>
Data & backend
State management
Networking & http
<b>Overview</b>

Even if you missed it, you can watch the recording of the December 17th livestream and Q&A!

Data & backend > Networking

# Networking

Internet network calls in Flutter.



## On this page

- Cross-platform http networking
- Platform notes
- Android
- macOS
- Samples

## Cross-platform http networking

The [http](#) package provides the simplest way to issue http requests. This package is supported on Android, iOS, macOS, Windows, Linux and the web.

## Platform notes

Some platforms require additional steps, as detailed below.

# Platform notes

Some platforms require additional steps, as detailed below.

## ต้องมีการขอ Permission ก่อน

### Android

Android apps must [declare their use of the internet](#) in the Android manifest (`AndroidManifest.xml`):

```
<manifest xmlns:android...>
  ...
  <uses-permission android:name="android.permission.INTERNET" />
  <application ...
</manifest>
```



### macOS

macOS apps must allow network access in the relevant `*.entitlements` files.

```
<key>com.apple.security.network.client</key>
<true/>
```

xml

Learn more about [setting up entitlements](#).

## Samples

For a practical sample of various networking tasks (incl. fetching data, WebSockets, and parsing data in the background) see the [networking cookbook recipes](#).

มาเริ่มดึง Data จาก Internet ไปพร้อมกัน

# Fetch data from the internet

How to fetch data over the internet using the `http` package.

Fetching data from the internet is necessary for most apps. Luckily, Dart and Flutter provide tools, such as the `http` package, for this type of work.

## Note

You should avoid directly using `dart:io` or `dart:html` to make HTTP requests. Those libraries are platform-dependent and tied to a single implementation.

This recipe uses the following steps:

1. Add the `http` package.
2. Make a network request using the `http` package.
3. Convert the response into a custom Dart object.
4. Fetch and display the data with Flutter.

# 1. Add the `http` package

The `http` package provides the simplest way to fetch data from the internet.

To add the `http` package as a dependency, run `flutter pub add`:

```
$ flutter pub add http
```

Import the `http` package.

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

dart

If you are deploying to Android, edit your `AndroidManifest.xml` file to add the Internet permission.

```
<!-- Required to fetch data from the internet. -->  
<uses-permission android:name="android.permission.INTERNET" />
```



Likewise, if you are deploying to macOS, edit your `macos/Runner/DebugProfile.entitlements` and `macos/Runner/Release.entitlements` files to include the network client entitlement.

```
<!-- Required to fetch data from the internet. -->  
<key>com.apple.security.network.client</key>  
<true/>
```

xml

## 2. Make a network request

This recipe covers how to fetch a sample album from the [JSONPlaceholder](#) using the `http.get()` method.

```
dart
Future<http.Response> fetchAlbum() {
  return http.get(Uri.parse('https://jsonplaceholder.typicode.com/albums/1'));
}
```

The `http.get()` method returns a `Future` that contains a `Response`.

- `Future` is a core Dart class for working with async operations. A Future object represents a potential value or error that will be available at some time in the future.
- The `http.Response` class contains the data received from a successful http call.

### 3. Convert the response into a custom Dart object

While it's easy to make a network request, working with a raw `Future<http.Response>` isn't very convenient. To make your life easier, convert the `http.Response` into a Dart object.

#### Create an `Album` class

First, create an `Album` class that contains the data from the network request. It includes a factory constructor that creates an `Album` from JSON.

Converting JSON using [pattern matching](#) is only one option. For more information, see the full article on [JSON and serialization](#).

```
dart

class Album {
    final int userId;
    final int id;
    final String title;

    const Album({required this.userId, required this.id, required this.title});

    factory Album.fromJson(Map<String, dynamic> json) {
        return switch (json) {
            {'userId': int userId, 'id': int id, 'title': String title} => Album(
                userId: userId,
                id: id,
                title: title,
            ),
            _ => throw const FormatException('Failed to load album.'),
        );
    }
}
```

## Convert the `http.Response` to an `Album`

Now, use the following steps to update the `fetchAlbum()` function to return a `Future<Album>`:

1. Convert the response body into a JSON `Map` with the `dart:convert` package.
2. If the server does return an OK response with a status code of 200, then convert the JSON `Map` into an `Album` using the `fromJson()` factory method.
3. If the server does not return an OK response with a status code of 200, then throw an exception. (Even in the case of a "404 Not Found" server response, throw an exception. Do not return `null`. This is important when examining the data in `snapshot`, as shown below.)

```
Future<Album> fetchAlbum() async {
    final response = await http.get(
        Uri.parse('https://jsonplaceholder.typicode.com/albums/1'),
    );

    if (response.statusCode == 200) {
        // If the server did return a 200 OK response,
        // then parse the JSON.
        return Album.fromJson(jsonDecode(response.body) as Map<String, dynamic>);
    } else {
        // If the server did not return a 200 OK response,
        // then throw an exception.
        throw Exception('Failed to load album');
    }
}
```

Hooray! Now you've got a function that fetches an album from the internet.

## 4. Fetch the data

Call the `fetchAlbum()` method in either the `initState()` or `didChangeDependencies()` methods.

The `initState()` method is called exactly once and then never again. If you want to have the option of reloading the API in response to an `InheritedWidget` changing, put the call into the `didChangeDependencies()` method. See [State](#) for more details.

```
dart
class _MyAppState extends State<MyApp> {
    late Future<Album> futureAlbum;

    @override
    void initState() {
        super.initState();
        futureAlbum = fetchAlbum();
    }
    // ...
}
```

This Future is used in the next step.

## 5. Display the data

To display the data on screen, use the `FutureBuilder` widget. The `FutureBuilder` widget comes with Flutter and makes it easy to work with asynchronous data sources.

You must provide two parameters:

1. The `Future` you want to work with. In this case, the future returned from the `fetchAlbum()` function.
2. A `builder` function that tells Flutter what to render, depending on the state of the `Future`: loading, success, or error.

Note that `snapshot.hasData` only returns `true` when the snapshot contains a non-null data value.

Because `fetchAlbum` can only return non-null values, the function should throw an exception even in the case of a "404 Not Found" server response. Throwing an exception sets the `snapshot.hasError` to `true` which can be used to display an error message.

Otherwise, the spinner will be displayed.

```
FutureBuilder<Album>(  
  future: futureAlbum,  
  builder: (context, snapshot) {  
    if (snapshot.hasData) {  
      return Text(snapshot.data!.title);  
    } else if (snapshot.hasError) {  
      return Text('${snapshot.error}');  
    }  
  
    // By default, show a loading spinner.  
    return const CircularProgressIndicator();  
  },  
)
```

# Why is `fetchAlbum()` called in `initState()`?

Although it's convenient, it's not recommended to put an API call in a `build()` method.

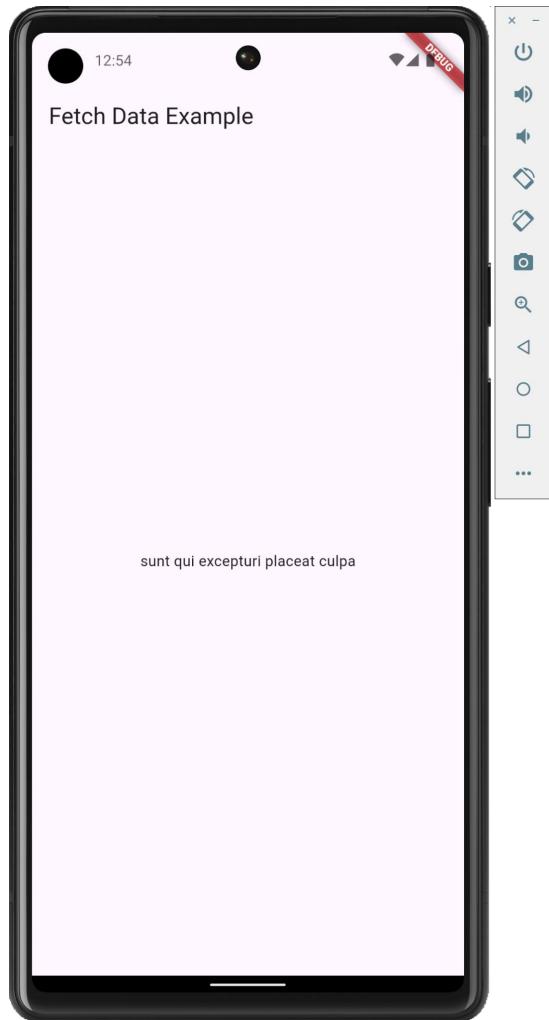
Flutter calls the `build()` method every time it needs to change anything in the view, and this happens surprisingly often. The `fetchAlbum()` method, if placed inside `build()`, is repeatedly called on each rebuild causing the app to slow down.

Storing the `fetchAlbum()` result in a state variable ensures that the `Future` is executed only once and then cached for subsequent rebuilds.

## Testing

For information on how to test this functionality, see the following recipes:

- [Introduction to unit testing](#)
- [Mock dependencies using Mockito](#)



w Go Run Terminal Window Help

1

New Terminal

^ ⌘ ⌂

Split Terminal

⌘ \

New Terminal Window

^ ⌘ ⌂

elc

[PROBLEMS](#)[OUTPUT](#)[TERMINAL](#)[...](#)

The default interactive shell is now  
To update your account to use zsh, pl  
For more details, please visit <https://www.burnto.dev/zsh>  
(base) Kittikorns-MacBook-Pro:flutter  
o flutter pub add http█

[PROBLEMS](#)[OUTPUT](#)[TERMINAL](#)

...



bash

RNT  
EV

```
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Kittikorns-MacBook-Pro:flutter_application_1 kittikornprasertsak$ ● flutter pub add http
Resolving dependencies...
Downloading packages...
  characters 1.4.0 (1.4.1 available)
+ http 1.6.0
+ http_parser 4.1.2
  matcher 0.12.17 (0.12.18 available)
  material_color_utilities 0.11.1 (0.13.0 available)
  test_api 0.7.7 (0.7.8 available)
+ typed_data 1.4.0
+ web 1.1.1
Changed 4 dependencies!
4 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
(base) Kittikorns-MacBook-Pro:flutter_application_1 kittikornprasertsak$
```

lib &gt; main.dart &gt; ...

```
1 import 'dart:async';
2 import 'dart:convert';
3
4 import 'package:flutter/material.dart';
5 import 'package:http/http.dart' as http;
6
7 Future<Album> fetchAlbum() async {
8     final response = await http.get(
9         Uri.parse('https://jsonplaceholder.typicode.com/albums/2'),
10    );
11
12     if (response.statusCode == 200) {
13         // If the server did return a 200 OK response,
14         // then parse the JSON.
15         return Album.fromJson(jsonDecode(response.body) as Map<String, dynamic>);
16     } else {
17         // If the server did not return a 200 OK response,
18         // then throw an exception.
19         throw Exception('Failed to load album');
20     }
21 }
```

lib &gt; main.dart &gt; ...

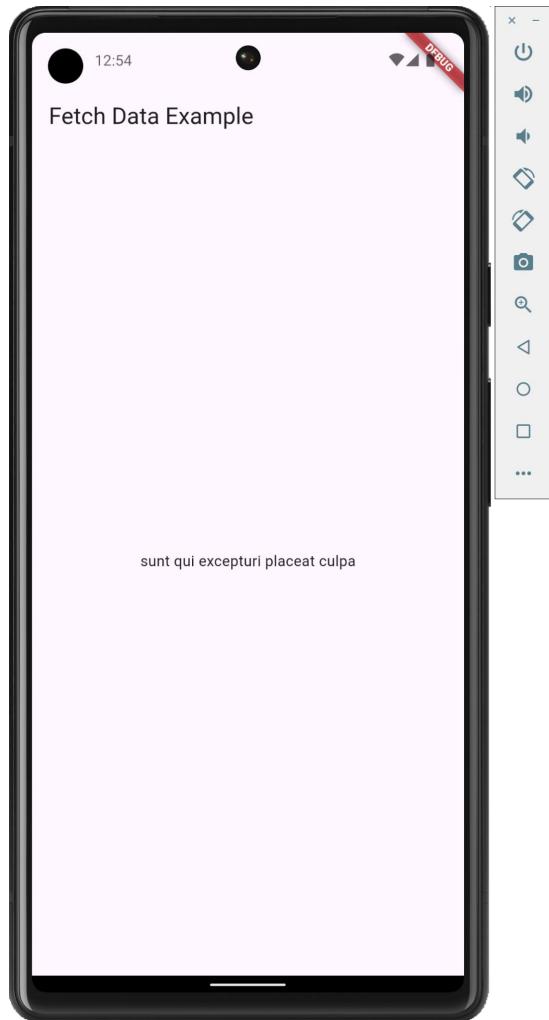
```
23 class Album {  
24     final int userId;  
25     final int id;  
26     final String title;  
27  
28     const Album({required this.userId, required this.id, required this.title});  
29  
30     factory Album.fromJson(Map<String, dynamic> json) {  
31         return switch (json) {  
32             {'userId': int userId, 'id': int id, 'title': String title} => Album(  
33                 userId: userId,  
34                 id: id,  
35                 title: title,  
36             ),  
37             _ => throw const FormatException('Failed to load album.'),  
38         };  
39     }  
40 }  
41 }
```

lib &gt; main.dart &gt; ...

Run | Debug | Profile

```
42 void main() => runApp(const MyApp());  
43  
44 class MyApp extends StatefulWidget {  
45     const MyApp({super.key});  
46  
47     @override  
48     State<MyApp> createState() => _MyAppState();  
49 }  
50  
51 class _MyAppState extends State<MyApp> {  
52     late Future<Album> futureAlbum;  
53  
54     @override  
55     void initState() {  
56         super.initState();  
57         futureAlbum = fetchAlbum();  
58     }
```

```
lib > main.dart > _MyAppState
51 class _MyAppState extends State<MyApp> {
52
53     @override
54     Widget build(BuildContext context) {
55         return MaterialApp(
56             title: 'Fetch Data Example',
57             theme: ThemeData(
58                 colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
59             ), // ThemeData
60             home: Scaffold(
61                 appBar: AppBar(title: const Text('Fetch Data Example')),
62                 body: Center(
63                     child: FutureBuilder<Album>(
64                         future: futureAlbum,
65                         builder: (context, snapshot) {
66                             if (snapshot.hasData) {
67                                 return Text(snapshot.data!.title);
68                             } else if (snapshot.hasError) {
69                                 return Text('${snapshot.error}');
70                             }
71
72                             // By default, show a loading spinner.
73                             return const CircularProgressIndicator();
74                         },
75                     ),
76                 ), // FutureBuilder
77             ), // Center
78         ), // Scaffold
79     ); // MaterialApp
80 }
81 }
```



# ตัวอย่างการทำ Authen Request

```
7 Future<Album> fetchAlbum() async {
8     final response = await http.get(
9         Uri.parse('https://jsonplaceholder.typicode.com/albums/1'),
10        // Send authorization headers to the backend.
11        headers: {HttpHeaders.authorizationHeader: 'Basic your_api_token_here'},
12    );
13    final responseJson = jsonDecode(response.body) as Map<String, dynamic>;
14
15    return Album.fromJson(responseJson);
16 }
```

Accessibility ▾

Internationalization

Beyond UI

Data & backend ▾

State management ▾

Networking & http ▾

Overview

Fetch data from the internet

[Make authenticated requests](#)

Send data to the internet

Update data over the internet

Delete data on the internet

Communicate with  
WebSockets

Serialization ▾

Persistence ▾

Firebase ▾

Google APIs

App architecture ▾

Even if you missed it, you can watch the recording of the December 17th livestream and Q&A!

Cookbook > Networking > Make authenticated requests



# Make authenticated requests

How to fetch authorized data from a web service.

To fetch data from most web services, you need to provide authorization. There are many ways to do this, but perhaps the most common uses the `Authorization` HTTP header.

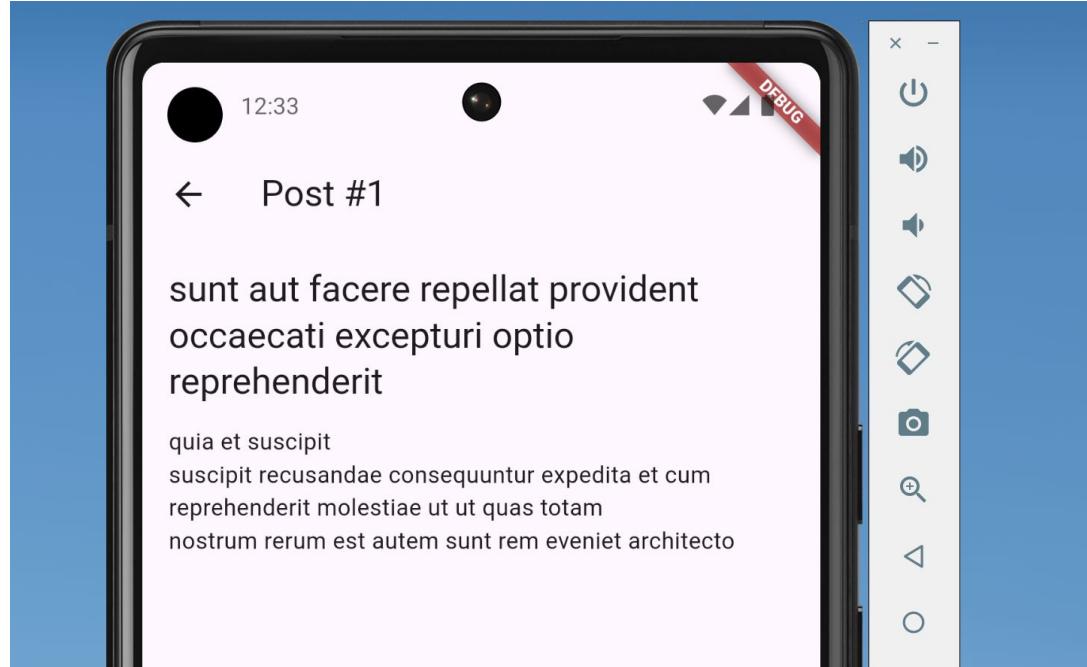
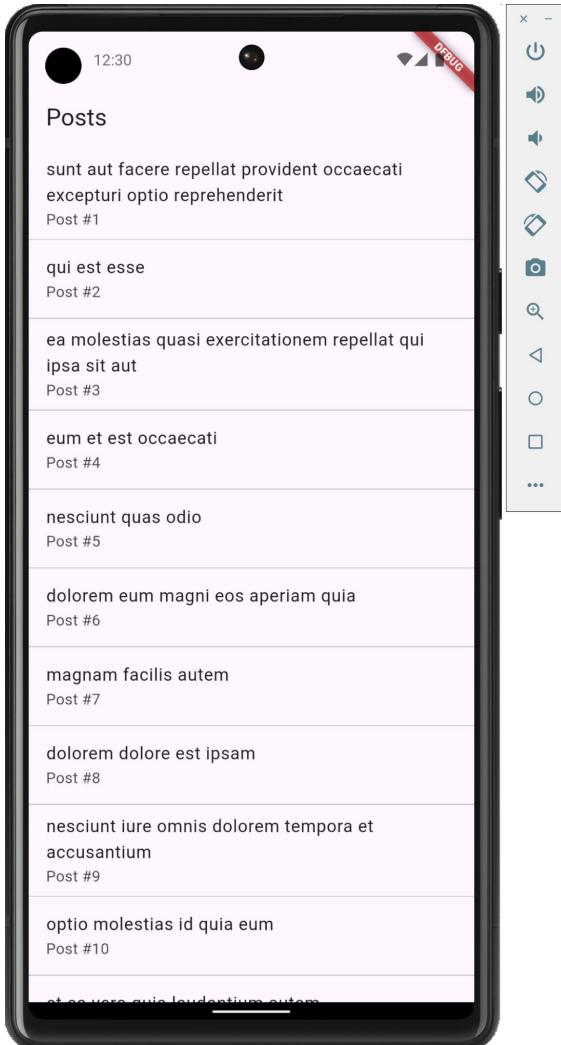
## Add authorization headers

The `http` package provides a convenient way to add headers to your requests. Alternatively, use the `HttpHeaders` class from the `dart:io` library.

```
final response = await http.get(  
  Uri.parse('https://jsonplaceholder.typicode.com/albums/1'),  
  // Send authorization headers to the backend.  
  headers: {HttpHeaders.authorizationHeader: 'Basic your_api_token_here'},  
);
```

dart

ลองดูตัวอย่างของตัวเองกัน



ปลายทางอย่างได้แบบนี้

w Go Run Terminal Window Help

1

New Terminal

^ ⌘ ⌂

Split Terminal

⌘ \

New Terminal Window

^ ⌘ ⌂

elc

[PROBLEMS](#)[OUTPUT](#)[TERMINAL](#)[...](#)

The default interactive shell is now  
To update your account to use zsh, pl  
For more details, please visit <https://www.zsh.org/>  
(base) Kittikorns-MacBook-Pro:flutter  
○ flutter pub add http█

[PROBLEMS](#)[OUTPUT](#)[TERMINAL](#)

...

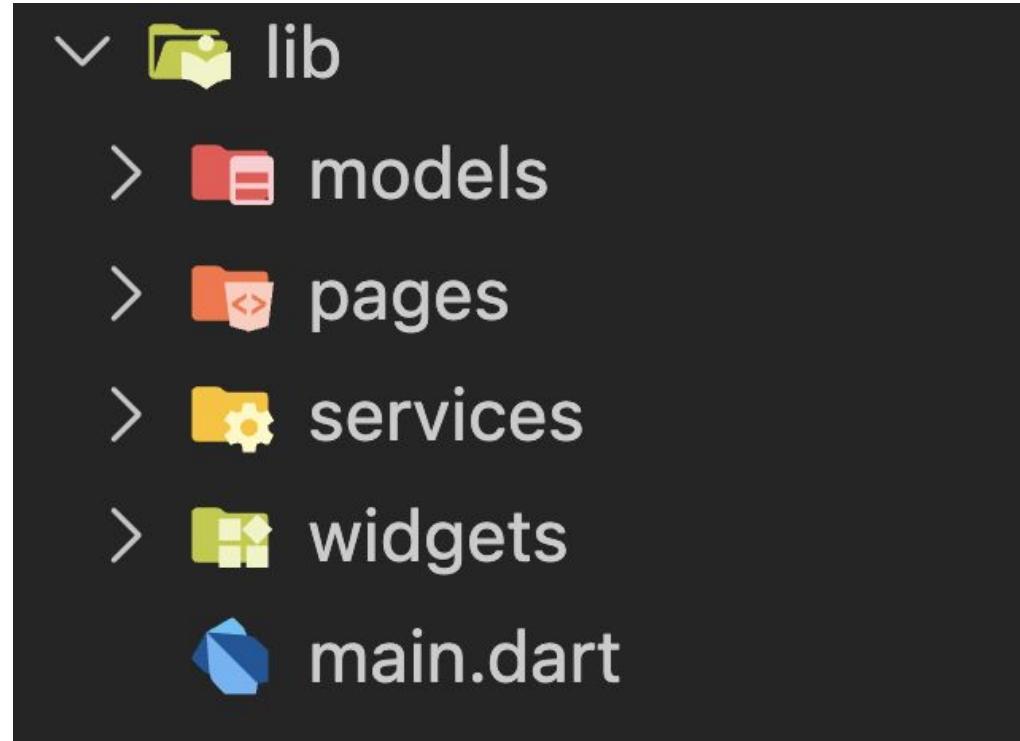


bash

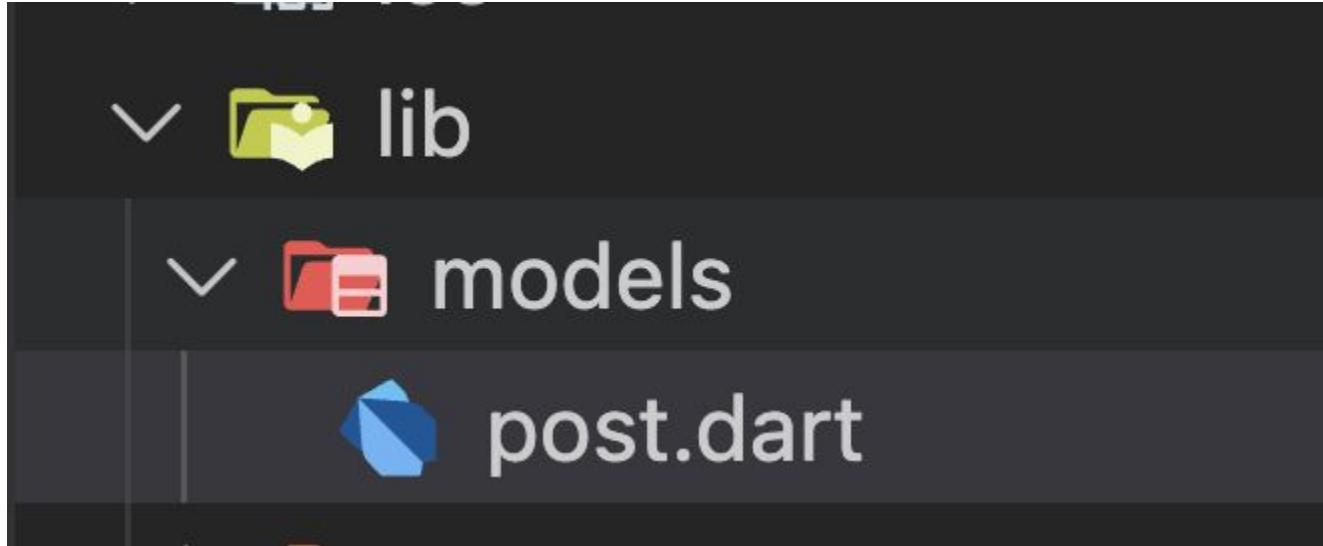


To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit <https://support.apple.com/kb/HT208050>.

(base) Kittikorns-MacBook-Pro:flutter\_application\_1 kittikornprasertsak\$ ● flutter pub add http  
Resolving dependencies...  
Downloading packages...  
  **characters** 1.4.0 (1.4.1 available)  
+ **http** 1.6.0  
+ **http\_parser** 4.1.2  
  **matcher** 0.12.17 (0.12.18 available)  
  **material\_color\_utilities** 0.11.1 (0.13.0 available)  
  **test\_api** 0.7.7 (0.7.8 available)  
+ **typed\_data** 1.4.0  
+ **web** 1.1.1  
Changed 4 dependencies!  
4 packages have newer versions incompatible with dependency constraints.  
Try `flutter pub outdated` for more information.  
(base) Kittikorns-MacBook-Pro:flutter\_application\_1 kittikornprasertsak\$



สร้างโฟลเดอร์ดังนี้ภายใต้ lib



สร้าง model ส่วน post.dart

A screenshot of a web browser displaying the JSONPlaceholder website at <https://jsonplaceholder.typicode.com>. The page features a blue header bar with the text "Check my new project 🌟 MistCSS write React components with 50% less code". Below the header, there's a navigation bar with links for "JSONPlaceholder", "Guide", "Sponsor this project", "Blog", and "My JSON Server".

JSONPlaceholder

Guide Sponsor this project Blog My JSON Server

# {JSON} Placeholder

Free fake and reliable API for testing and prototyping.

Powered by [JSON Server](#) + [LowDB](#).

Serving ~3 billion requests each month.

ตัวอย่างข้อมูลที่เราจะไปทำการดึงมาแสดง  
<https://jsonplaceholder.typicode.com/>

```
[  
  {  
    "userId": 1,  
    "id": 1,  
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",  
    "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"  
  },  
  {  
    "userId": 1,  
    "id": 2,  
    "title": "qui est esse",  
    "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis possimus qui neque nisi nulla"  
  },  
  {  
    "userId": 1,  
    "id": 3,  
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",  
    "body": "et iusto sed quo iure\\nvolutatem occaecati omnis eligendi aut ad\\nvolutatem doloribus vel accusantium quis pariatur\\nmolestiae porro eius odio et labore et velit aut"  
  },  
  {  
    "userId": 1,  
    "id": 4,  
    "title": "eum et est occaecati",  
    "body": "ullam et saepe reiciendis voluptatem adipisci\\nsit amet autem assumenda provident rerum culpa\\nquis hic commodi nesciunt rem tenetur doloremeque ipsam iure\\nquis sunt voluptatem rerum illo velit"  
  },  
  {  
    "userId": 1,  
    "id": 5,  
    "title": "nesciunt quas odio",  
    "body": "repudiandae veniam quaerat sunt sed\\nalias aut fugiat sit autem sed est\\nvolutatem omnis possimus esse voluptatibus quis\\nest aut tenetur dolor neque"  
  },  
  {  
    "userId": 1,  
    "id": 6,  
    "title": "dolorem eum magni eos aperiam quia",  
    "body": "ut aspernatur corporis harum nihil quis provident sequi\\nmollitia nobis aliquid molestiae\\nperspiciatis et ea nemo ab reprehenderit accusantium quas\\nvolutate dolores velit et doloremeque molestiae"  
  },  
  {  
    "userId": 1,  
    "id": 7,  
    "title": "magnum facilis autem",  
    "body": "dolore placeat quibusdam ea quo vitae\\nmagni quis enim qui quis quo nemo aut saepe\\nquidem repellat excepturi ut quia\\nsunt ut sequi eos ea sed quas"  
  },  
  {  
    "userId": 1,  
    "id": 8,  
    "title": "dolorem dolore est ipsam",  
    "body": "dignissimos aperiam dolorem qui eum\\nfacilis quibusdam animi sint suscipit qui sint possimus cum\\nquaerat magni maiores excepturi\\nipsam ut commodi dolor voluptatum modi aut vitae"  
},
```

<https://jsonplaceholder.typicode.com/posts>

```
[  
 {  
   "userId": 1,  
   "id": 1,  
   "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",  
   "body": "quia et suscipit\\nsuscipit recusandae consequuntur expedita et cum\\nreprehenderit molestiae  
ut ut quas totam\\nnostrum rerum est autem sunt rem eveniet architecto"  
 },  
 {  
   "userId": 1,  
   "id": 2,  
   "title": "qui est esse",  
   "body": "est rerum tempore vitae\\nsequi sint nihil reprehenderit dolor beatae ea dolores  
neque\\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\\nqui aperiam non debitis  
possimus qui neque nisi nulla"  
 },  
 {  
   "userId": 1,  
   "id": 3,  
   "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",  
   "body": "et iusto sed quo iure\\nvoluptatem occaecati omnis eligendi aut ad\\nvoluptatem doloribus vel  
accusantium quis pariatur\\nmolestiae porro eius odio et labore et velit aut"  
 },
```

<https://jsonplaceholder.typicode.com/posts>

lib &gt; models &gt; post.dart &gt; ...

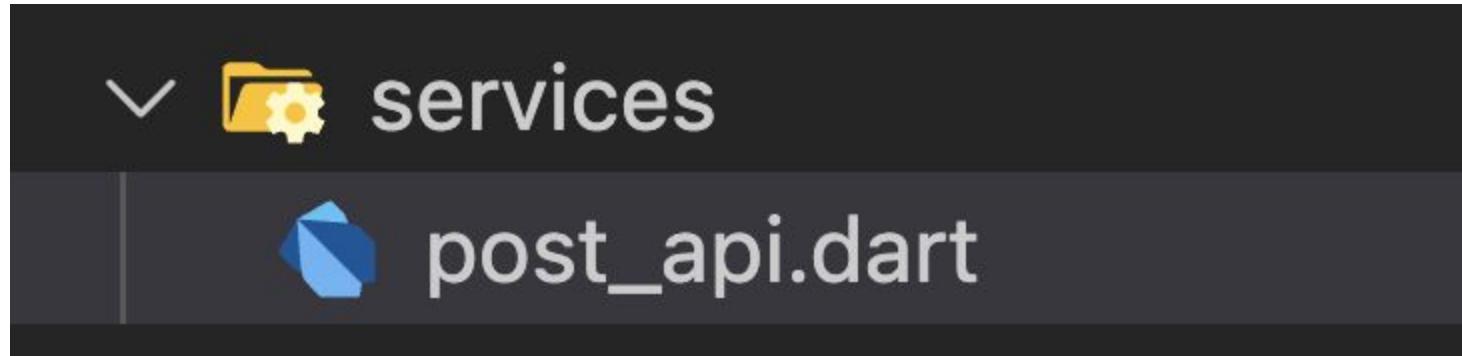
```
1  class Post {  
2      final int id;  
3      final int userId;  
4      final String title;  
5      final String body;  
6  
7      Post({  
8          required this.id,  
9          required this.userId,  
10         required this.title,  
11         required this.body,  
12     });  
13 }
```

สร้าง Model ใน post.dart ให้เรียบร้อย



```
lib > models > post.dart > Post
1   class Post {
2
3
4       factory Post.fromJson(Map<String, dynamic> json) {
5           return Post(
6               id: json['id'] as int,
7               userId: json['userId'] as int,
8               title: json['title'] as String,
9               body: json['body'] as String,
10          );
11      }
12
13
14 }
```

ทำการ Decode อกมาเป็น Map / List เพื่อจัดการต่อได้

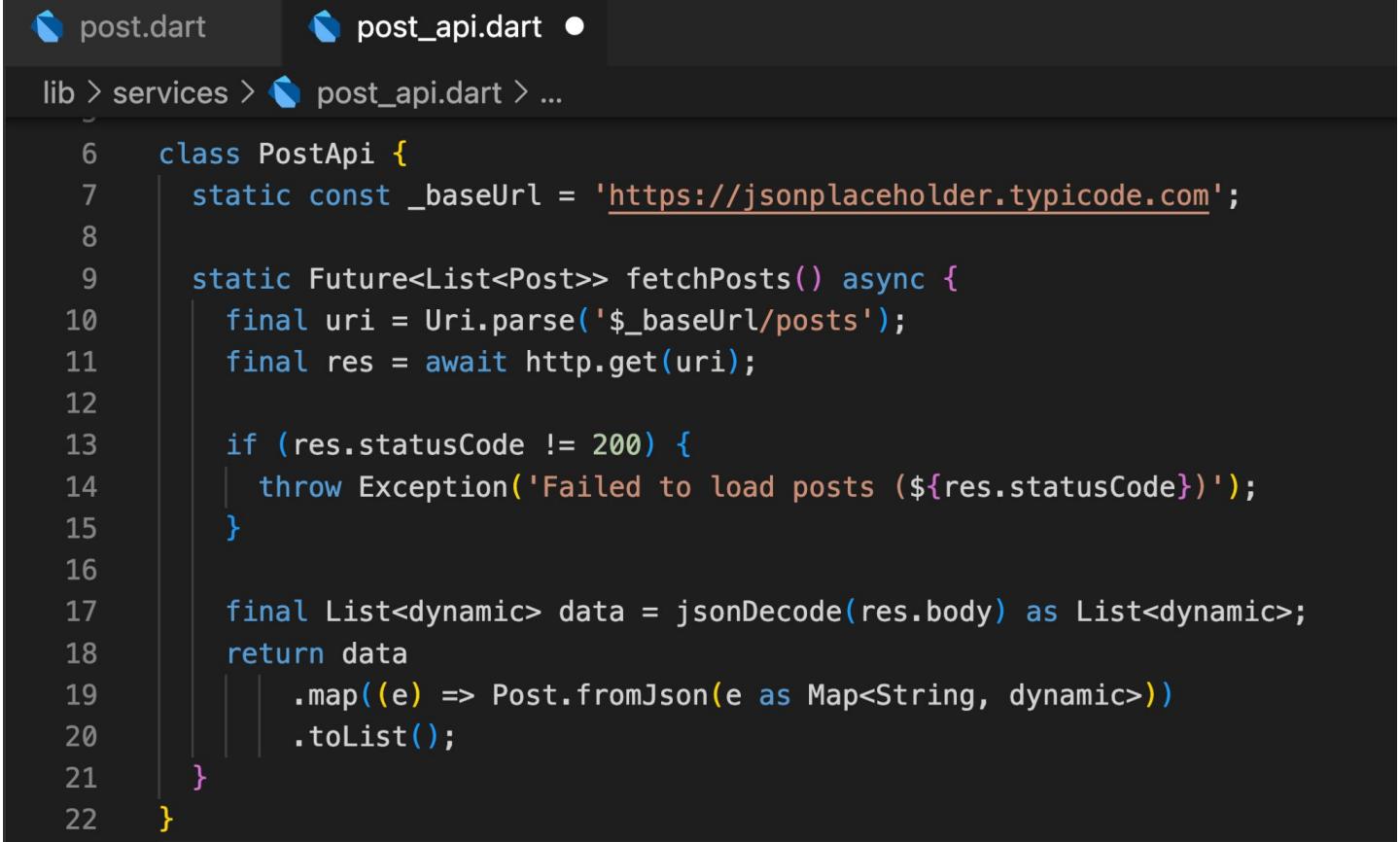


สร้างไฟล์ post\_api.dart

 post.dart post\_api.dart 3lib > services >  post\_api.dart

```
1 import 'dart:convert';
2 import 'package:http/http.dart' as http;
3
4 import '../models/post.dart';
```

เริ่มจาก import ที่จำเป็น



The screenshot shows a code editor with two tabs at the top: "post.dart" and "post\_api.dart". The "post\_api.dart" tab is active, displaying the following Dart code:

```
6  class PostApi {
7      static const _baseUrl = 'https://jsonplaceholder.typicode.com';
8
9      static Future<List<Post>> fetchPosts() async {
10         final uri = Uri.parse('${_baseUrl}/posts');
11         final res = await http.get(uri);
12
13         if (res.statusCode != 200) {
14             throw Exception('Failed to load posts (${res.statusCode})');
15         }
16
17         final List<dynamic> data = jsonDecode(res.body) as List<dynamic>;
18         return data
19             .map((e) => Post.fromJson(e as Map<String, dynamic>))
20             .toList();
21     }
22 }
```

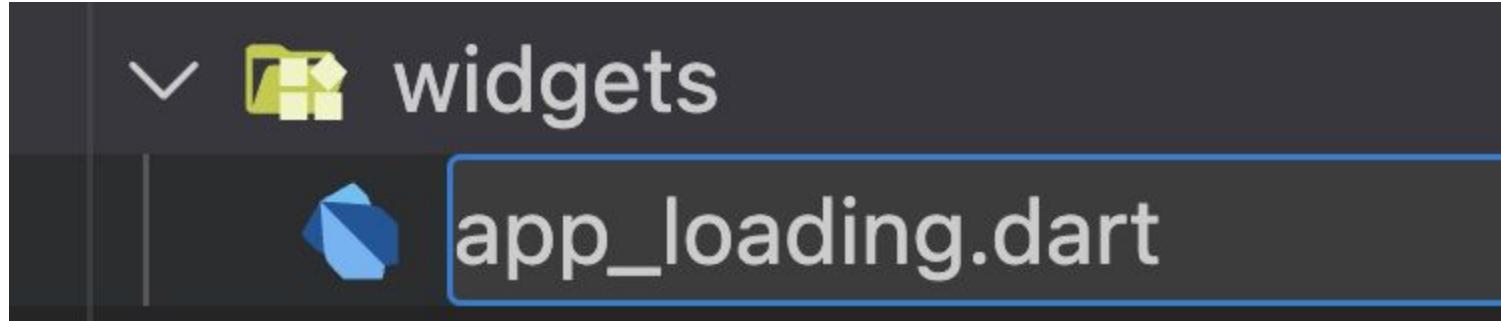
ทำเมธอดดึง “รายการโพสต์”

post.dart

post\_api.dart ●

```
lib > services > post_api.dart > ...
6   class PostApi {
23     static Future<Post> fetchPostDetail(int id) async {
24       final uri = Uri.parse('${_baseUrl}/posts/$id');
25       final res = await http.get(uri);
26
27       if (res.statusCode != 200) {
28         throw Exception('Failed to load post ($id) (${res.statusCode})');
29       }
30
31       final Map<String, dynamic> data =
32         jsonDecode(res.body) as Map<String, dynamic>;
33       return Post.fromJson(data);
34     }
35
36 }
```

ทำเมธอดดึง “รายละเอียดโพสต์”



สร้างไฟล์ app\_loading.dart ไว้ใน widgets



post.dart

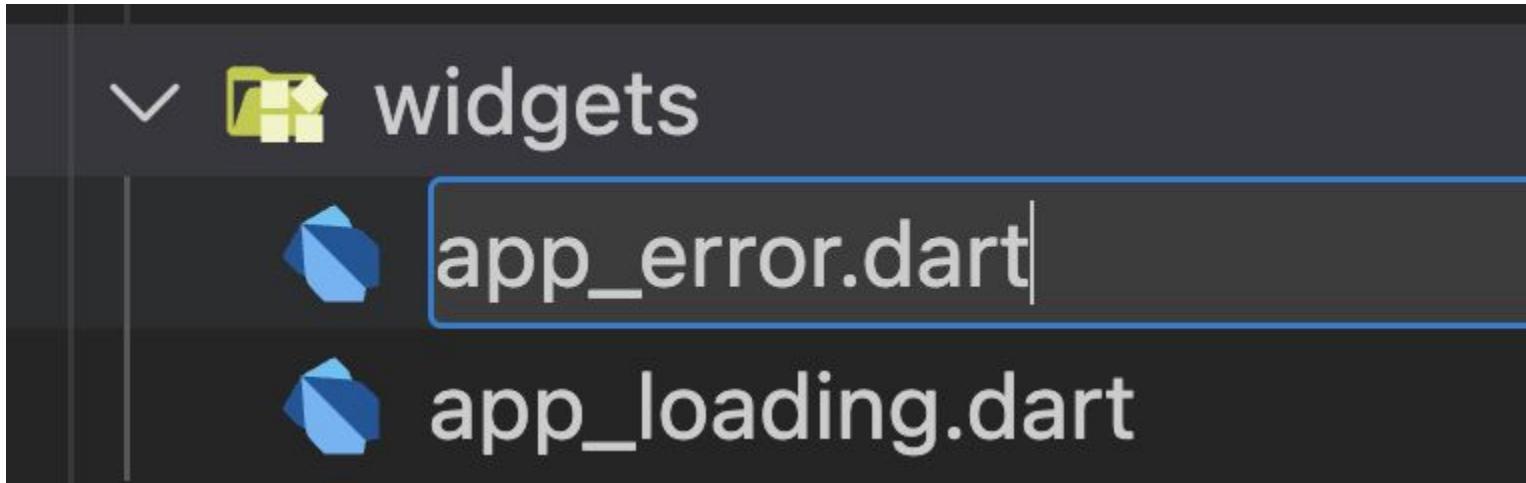
post\_api.dart

app\_loading.dart ×

lib &gt; widgets &gt; app\_loading.dart &gt; ...

```
1 import 'package:flutter/material.dart';
2
3 class AppLoading extends StatelessWidget {
4     const AppLoading({super.key});
5
6     @override
7     Widget build(BuildContext context) {
8         return const Center(child: CircularProgressIndicator());
9     }
10 }
```

สร้างไฟล์ app\_loading.dart ไว้ใน widgets



สร้างไฟล์ app\_error.dart ไว้ใน widgets

post\_api.dart app\_loading.dart app\_error.dart

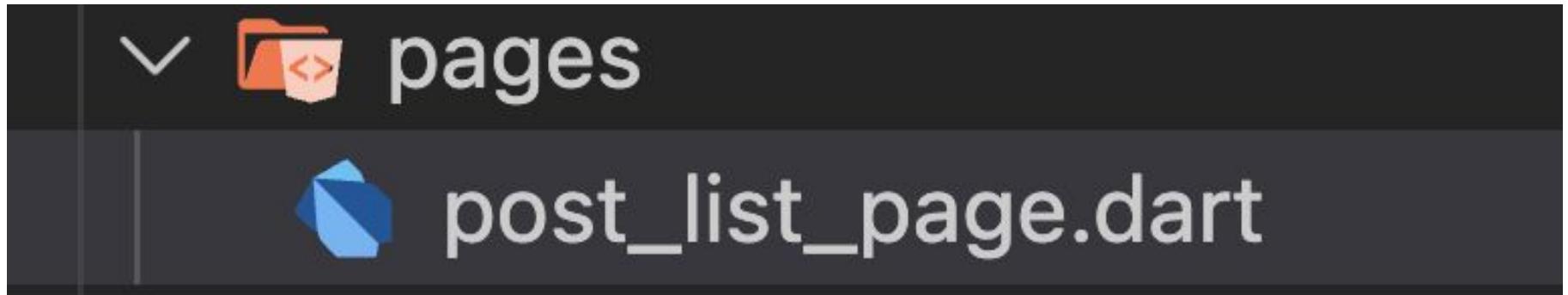
```
lib > widgets > app_error.dart > ...
1 import 'package:flutter/material.dart';
2 Q
3 class AppError extends StatelessWidget {
4   final Object error;
5   final VoidCallback? onRetry;
6
7   const AppError({
8     super.key,
9     required this.error,
10    this.onRetry,
11  });
12
13   @override
14   Widget build(BuildContext context) {
15     return Center(
16       );
17   }
18 }
19 }
```

สร้างไฟล์ app\_error.dart ไว้ใน widgets

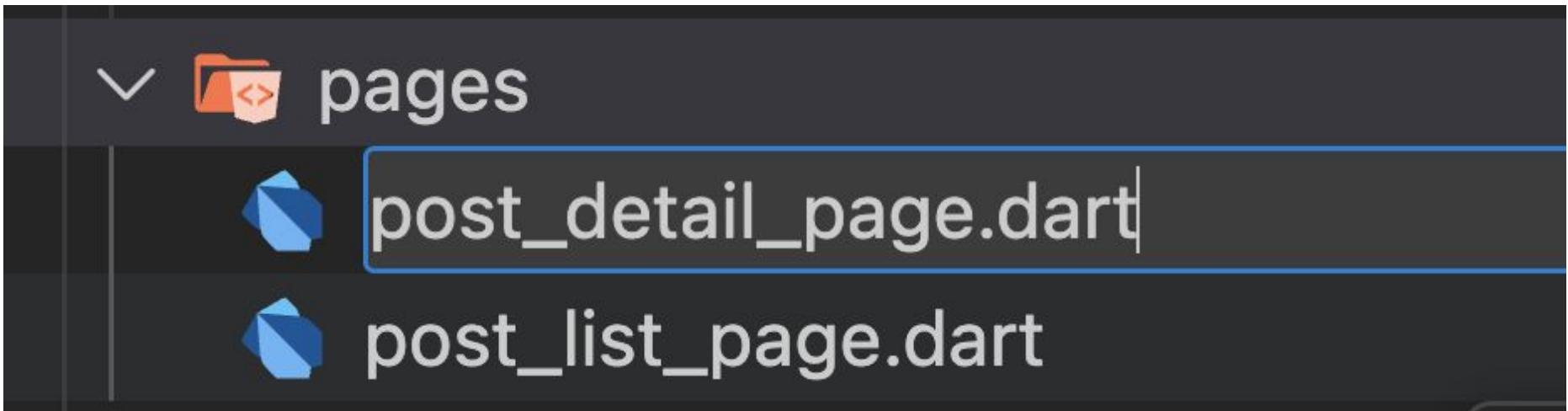
The screenshot shows a code editor with a dark theme. At the top, there are three tabs: 'post\_api.dart', 'app\_loading.dart', and 'app\_error.dart'. The 'app\_error.dart' tab is active. The code is as follows:

```
lib > widgets > app_error.dart > AppError
  3   class AppError extends StatelessWidget {
  4
  5     @override
  6     Widget build(BuildContext context) {
  7       return Center(
  8         child: Padding(
  9           padding: const EdgeInsets.all(16),
 10           child: Column(
 11             mainAxisSize: MainAxisSize.min,
 12             children: [
 13               Text('✖ $error'),
 14               const SizedBox(height: 12),
 15               if (onRetry != null)
 16                 ElevatedButton(
 17                   onPressed: onRetry,
 18                   child: const Text('Retry'),
 19                 ), // ElevatedButton
 20               ],
 21             ), // Column
 22           ), // Padding
 23         ); // Center
 24     }
 25   }
```

สร้างไฟล์ app\_error.dart ไว้ใน widgets



สร้างไฟล์ post\_list\_page.dart ไว้ใน pages



สร้างไฟล์ post\_detail\_page.dart ไว้ใน Pages

```
lib > pages > post_list_page.dart > ...
1   import 'package:flutter/material.dart';
2
3   import '../models/post.dart';
4   import '../services/post_api.dart';
5   import '../widgets/app_error.dart';
6   import '../widgets/app_loading.dart';
7   import 'post_detail_page.dart';
8
9   class PostListPage extends StatefulWidget {
10     const PostListPage({super.key});
11
12     @override
13     State<PostListPage> createState() => _PostListPageState();
14   }
15
16   class _PostListPageState extends State<PostListPage> {
17     late Future<List<Post>> _futurePosts;
18
19     @override
20     void initState() {
21       super.initState();
22       _futurePosts = PostApi.fetchPosts();
23     }
}
```

สร้างไฟล์ post\_list\_page.dart ไว้ใน pages

lib > pages > 📄 post\_list\_page.dart > ...

```
16  class _PostListPageState extends State<PostListPage> {  
25  @override  
26  Widget build(BuildContext context) {  
27  return Scaffold(  
28  appBar: AppBar(title: const Text('Posts')),  
29  body: FutureBuilder<List<Post>>(  
30  future: _futurePosts,  
31  builder: (context, snapshot) {  
32  // 1) Loading  
33  if (snapshot.connectionState == ConnectionState.waiting) {  
34  return const AppLoading();  
35  }  
36  }  
37  }  
38  }  
39  }  
40  }  
41  }  
42  }  
43  }  
44  }  
45  }  
46  }  
47  }  
48  }  
49  }  
50  }  
51  }  
52  }  
53  }  
54  }  
55  }  
56  }  
57  }  
58  }  
59  }  
60  }  
61  }  
62  }  
63  }  
64  }  
65  }  
66  }  
67  }  
68  }  
69  }  
70  }  
71  }  
72  }  
73  }  
74  }  
75  }  
76  }  
77  }  
78  }  
79  }  
80  }  
81  }  
82  }  
83  }  
84  }  
85  }  
86  }  
87  }  
88  }  
89  }  
90  }  
91  }  
92  }  
93  }  
94  }  
95  }  
96  }  
97  }  
98  }  
99  }  
100 }  
101 }  
102 }  
103 }  
104 }  
105 }  
106 }  
107 }  
108 }  
109 }  
110 }  
111 }  
112 }  
113 }  
114 }  
115 }  
116 }  
117 }  
118 }  
119 }  
120 }  
121 }  
122 }  
123 }  
124 }  
125 }  
126 }  
127 }  
128 }  
129 }  
130 }  
131 }  
132 }  
133 }  
134 }  
135 }  
136 }  
137 }  
138 }  
139 }  
140 }  
141 }  
142 }  
143 }  
144 }  
145 }  
146 }  
147 }  
148 }  
149 }  
150 }  
151 }  
152 }  
153 }  
154 }  
155 }  
156 }  
157 }  
158 }  
159 }  
160 }  
161 }  
162 }  
163 }  
164 }  
165 }  
166 }  
167 }  
168 }  
169 }  
170 }  
171 }  
172 }  
173 }  
174 }  
175 }  
176 }  
177 }  
178 }  
179 }  
180 }  
181 }  
182 }  
183 }  
184 }  
185 }  
186 }  
187 }  
188 }  
189 }  
190 }  
191 }  
192 }  
193 }  
194 }  
195 }  
196 }  
197 }  
198 }  
199 }  
200 }  
201 }  
202 }  
203 }  
204 }  
205 }  
206 }  
207 }  
208 }  
209 }  
210 }  
211 }  
212 }  
213 }  
214 }  
215 }  
216 }  
217 }  
218 }  
219 }  
220 }  
221 }  
222 }  
223 }  
224 }  
225 }  
226 }  
227 }  
228 }  
229 }  
230 }  
231 }  
232 }  
233 }  
234 }  
235 }  
236 }  
237 }  
238 }  
239 }  
240 }  
241 }  
242 }  
243 }  
244 }  
245 }  
246 }  
247 }  
248 }  
249 }  
250 }  
251 }  
252 }  
253 }  
254 }  
255 }  
256 }  
257 }  
258 }  
259 }  
260 }  
261 }  
262 }  
263 }  
264 }  
265 }  
266 }  
267 }  
268 }  
269 }  
270 }  
271 }  
272 }  
273 }  
274 }  
275 }  
276 }  
277 }  
278 }  
279 }  
280 }  
281 }  
282 }  
283 }  
284 }  
285 }  
286 }  
287 }  
288 }  
289 }  
290 }  
291 }  
292 }  
293 }  
294 }  
295 }  
296 }  
297 }  
298 }  
299 }  
300 }  
301 }  
302 }  
303 }  
304 }  
305 }  
306 }  
307 }  
308 }  
309 }  
310 }  
311 }  
312 }  
313 }  
314 }  
315 }  
316 }  
317 }  
318 }  
319 }  
320 }  
321 }  
322 }  
323 }  
324 }  
325 }  
326 }  
327 }  
328 }  
329 }  
330 }  
331 }  
332 }  
333 }  
334 }  
335 }  
336 }  
337 }  
338 }  
339 }  
340 }  
341 }  
342 }  
343 }  
344 }  
345 }  
346 }  
347 }  
348 }  
349 }  
350 }  
351 }  
352 }  
353 }  
354 }  
355 }  
356 }  
357 }  
358 }  
359 }  
360 }  
361 }  
362 }  
363 }  
364 }  
365 }  
366 }  
367 }  
368 }  
369 }  
370 }  
371 }  
372 }  
373 }  
374 }  
375 }  
376 }  
377 }  
378 }  
379 }  
380 }  
381 }  
382 }  
383 }  
384 }  
385 }  
386 }  
387 }  
388 }  
389 }  
390 }  
391 }  
392 }  
393 }  
394 }  
395 }  
396 }  
397 }  
398 }  
399 }  
400 }  
401 }  
402 }  
403 }  
404 }  
405 }  
406 }  
407 }  
408 }  
409 }  
410 }  
411 }  
412 }  
413 }  
414 }  
415 }  
416 }  
417 }  
418 }  
419 }  
420 }  
421 }  
422 }  
423 }  
424 }  
425 }  
426 }  
427 }  
428 }  
429 }  
430 }  
431 }  
432 }  
433 }  
434 }  
435 }  
436 }  
437 }  
438 }  
439 }  
440 }  
441 }  
442 }  
443 }  
444 }  
445 }  
446 }  
447 }  
448 }  
449 }  
450 }  
451 }  
452 }  
453 }  
454 }  
455 }  
456 }  
457 }  
458 }  
459 }  
460 }  
461 }  
462 }  
463 }  
464 }  
465 }  
466 }  
467 }  
468 }  
469 }  
470 }  
471 }  
472 }  
473 }  
474 }  
475 }  
476 }  
477 }  
478 }  
479 }  
480 }  
481 }  
482 }  
483 }  
484 }  
485 }  
486 }  
487 }  
488 }  
489 }  
490 }  
491 }  
492 }  
493 }  
494 }  
495 }  
496 }  
497 }  
498 }  
499 }  
500 }  
501 }  
502 }  
503 }  
504 }  
505 }  
506 }  
507 }  
508 }  
509 }  
510 }  
511 }  
512 }  
513 }  
514 }  
515 }  
516 }  
517 }  
518 }  
519 }  
520 }  
521 }  
522 }  
523 }  
524 }  
525 }  
526 }  
527 }  
528 }  
529 }  
530 }  
531 }  
532 }  
533 }  
534 }  
535 }  
536 }  
537 }  
538 }  
539 }  
540 }  
541 }  
542 }  
543 }  
544 }  
545 }  
546 }  
547 }  
548 }  
549 }  
550 }  
551 }  
552 }  
553 }  
554 }  
555 }  
556 }  
557 }  
558 }  
559 }  
560 }  
561 }  
562 }  
563 }  
564 }  
565 }  
566 }  
567 }  
568 }  
569 }  
570 }  
571 }  
572 }  
573 }  
574 }  
575 }  
576 }  
577 }  
578 }  
579 }  
580 }  
581 }  
582 }  
583 }  
584 }  
585 }  
586 }  
587 }  
588 }  
589 }  
590 }  
591 }  
592 }  
593 }  
594 }  
595 }  
596 }  
597 }  
598 }  
599 }  
600 }  
601 }  
602 }  
603 }  
604 }  
605 }  
606 }  
607 }  
608 }  
609 }  
610 }  
611 }  
612 }  
613 }  
614 }  
615 }  
616 }  
617 }  
618 }  
619 }  
620 }  
621 }  
622 }  
623 }  
624 }  
625 }  
626 }  
627 }  
628 }  
629 }  
630 }  
631 }  
632 }  
633 }  
634 }  
635 }  
636 }  
637 }  
638 }  
639 }  
640 }  
641 }  
642 }  
643 }  
644 }  
645 }  
646 }  
647 }  
648 }  
649 }  
650 }  
651 }  
652 }  
653 }  
654 }  
655 }  
656 }  
657 }  
658 }  
659 }  
660 }  
661 }  
662 }  
663 }  
664 }  
665 }  
666 }  
667 }  
668 }  
669 }  
670 }  
671 }  
672 }  
673 }  
674 }  
675 }  
676 }  
677 }  
678 }  
679 }  
680 }  
681 }  
682 }  
683 }  
684 }  
685 }  
686 }  
687 }  
688 }  
689 }  
690 }  
691 }  
692 }  
693 }  
694 }  
695 }  
696 }  
697 }  
698 }  
699 }  
700 }  
701 }  
702 }  
703 }  
704 }  
705 }  
706 }  
707 }  
708 }  
709 }  
710 }  
711 }  
712 }  
713 }  
714 }  
715 }  
716 }  
717 }  
718 }  
719 }  
720 }  
721 }  
722 }  
723 }  
724 }  
725 }  
726 }  
727 }  
728 }  
729 }  
730 }  
731 }  
732 }  
733 }  
734 }  
735 }  
736 }  
737 }  
738 }  
739 }  
740 }  
741 }  
742 }  
743 }  
744 }  
745 }  
746 }  
747 }  
748 }  
749 }  
750 }  
751 }  
752 }  
753 }  
754 }  
755 }  
756 }  
757 }  
758 }  
759 }  
760 }  
761 }  
762 }  
763 }  
764 }  
765 }  
766 }  
767 }  
768 }  
769 }  
770 }  
771 }  
772 }  
773 }  
774 }  
775 }  
776 }  
777 }  
778 }  
779 }  
780 }  
781 }  
782 }  
783 }  
784 }  
785 }  
786 }  
787 }  
788 }  
789 }  
790 }  
791 }  
792 }  
793 }  
794 }  
795 }  
796 }  
797 }  
798 }  
799 }  
800 }  
801 }  
802 }  
803 }  
804 }  
805 }  
806 }  
807 }  
808 }  
809 }  
810 }  
811 }  
812 }  
813 }  
814 }  
815 }  
816 }  
817 }  
818 }  
819 }  
820 }  
821 }  
822 }  
823 }  
824 }  
825 }  
826 }  
827 }  
828 }  
829 }  
830 }  
831 }  
832 }  
833 }  
834 }  
835 }  
836 }  
837 }  
838 }  
839 }  
840 }  
841 }  
842 }  
843 }  
844 }  
845 }  
846 }  
847 }  
848 }  
849 }  
850 }  
851 }  
852 }  
853 }  
854 }  
855 }  
856 }  
857 }  
858 }  
859 }  
860 }  
861 }  
862 }  
863 }  
864 }  
865 }  
866 }  
867 }  
868 }  
869 }  
870 }  
871 }  
872 }  
873 }  
874 }  
875 }  
876 }  
877 }  
878 }  
879 }  
880 }  
881 }  
882 }  
883 }  
884 }  
885 }  
886 }  
887 }  
888 }  
889 }  
890 }  
891 }  
892 }  
893 }  
894 }  
895 }  
896 }  
897 }  
898 }  
899 }  
900 }  
901 }  
902 }  
903 }  
904 }  
905 }  
906 }  
907 }  
908 }  
909 }  
910 }  
911 }  
912 }  
913 }  
914 }  
915 }  
916 }  
917 }  
918 }  
919 }  
920 }  
921 }  
922 }  
923 }  
924 }  
925 }  
926 }  
927 }  
928 }  
929 }  
930 }  
931 }  
932 }  
933 }  
934 }  
935 }  
936 }  
937 }  
938 }  
939 }  
940 }  
941 }  
942 }  
943 }  
944 }  
945 }  
946 }  
947 }  
948 }  
949 }  
950 }  
951 }  
952 }  
953 }  
954 }  
955 }  
956 }  
957 }  
958 }  
959 }  
960 }  
961 }  
962 }  
963 }  
964 }  
965 }  
966 }  
967 }  
968 }  
969 }  
970 }  
971 }  
972 }  
973 }  
974 }  
975 }  
976 }  
977 }  
978 }  
979 }  
980 }  
981 }  
982 }  
983 }  
984 }  
985 }  
986 }  
987 }  
988 }  
989 }  
990 }  
991 }  
992 }  
993 }  
994 }  
995 }  
996 }  
997 }  
998 }  
999 }
```

สร้างไฟล์ post\_list\_page.dart ไว้ใน pages

```
lib > pages > post_list_page.dart > ...
16     class _PostListPageState extends State<PostListPage> {
26         Widget build(BuildContext context) {
30
37             // 2) Error
38             if (snapshot.hasError) {
39                 return AppError(
40                     error: snapshot.error!,
41                     onRetry: () {
42                         setState(() {
43                             _futurePosts = PostApi.fetchPosts();
44                         });
45                     },
46                     );
47             }
48         }
49     }
50 }
```

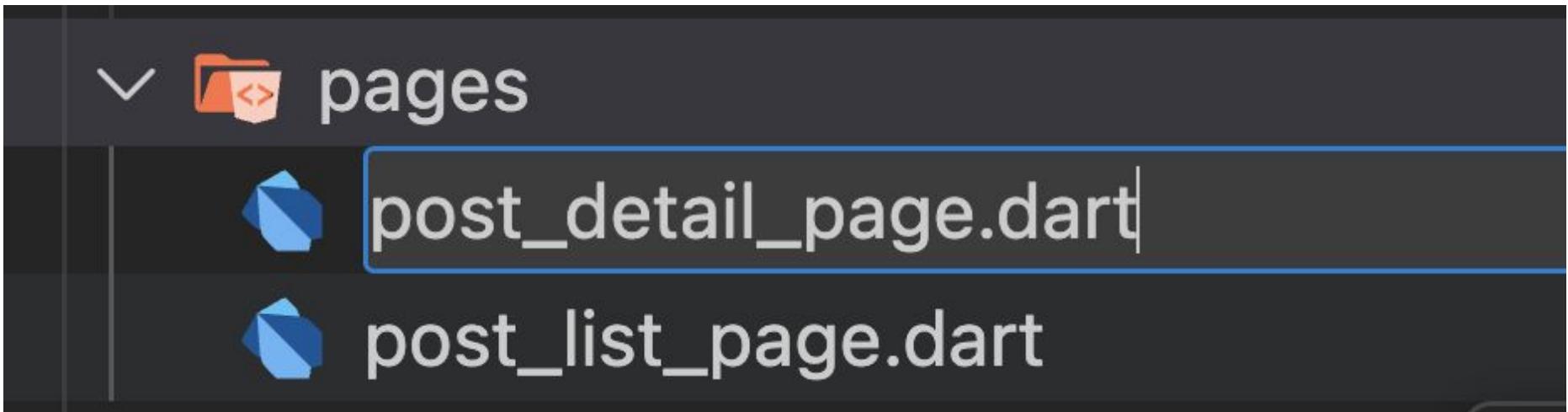
# สร้างไฟล์ post\_list\_page.dart ไว้ใน pages

```
16     class _PostListPageState extends State<PostListPage> {
26         Widget build(BuildContext context) {
40
49             // 3) Data
50             final posts = snapshot.data ?? [];
51             return ListView.separated(
52                 itemCount: posts.length,
53                 separatorBuilder: (_, __) => const Divider(height: 1),
54                 itemBuilder: (context, index) {
55                     final p = posts[index];
56                     return ListTile(
57                         title: Text(p.title),
58                         subtitle: Text('Post #${p.id}'),
59                         onTap: () {
60                             Navigator.push(
61                                 context,
62                                 MaterialPageRoute(
63                                     builder: (_) => PostDetailPage(postId: p.id),
64                                     ), // MaterialPageRoute
65                             );
66                         },
67                     ); // ListTile
68                 },
69             ); // ListView.separated
70         },

```

lib > pages > post\_list\_page.dart > ...

```
16     class _PostListPageState extends State<PostListPage> {
26         Widget build(BuildContext context) {
31             return Scaffold(
32                 appBar: AppBar(
33                     title: Text('Post List'),
34                 ),
35                 body: FutureBuilder(
36                     future: fetchPosts(),
37                     builder: (context, AsyncSnapshot snapshot) {
38                         if (snapshot.hasError) {
39                             return Text('Error: ${snapshot.error}');
40                         }
41                         return ListView.separated(
42                             itemCount: snapshot.data.length,
43                             itemBuilder: (context, index) {
44                                 Post post = snapshot.data[index];
45                                 return ListTile(
46                                     title: Text(post.title),
47                                     subtitle: Text(post.body),
48                                     trailing: IconButton(
49                                         icon: Icon(Icons.arrow_forward),
50                                         onPressed: () {
51                                             Navigator.push(
52                                                 context,
53                                                 MaterialPageRoute(
54                                                     builder: (context) => PostDetailPage(postId: post.id),
55                                                 ),
56                                             );
57                                         },
58                                     );
59                                 );
60                             },
61                             separatorBuilder: (context, index) {
62                                 return SizedBox(
63                                     height: 10.0,
64                                 );
65                             },
66                         );
67                     },
68                 ),
69             );
70         }
71     }
72 }
73 }
74 }
75 }
```



มาดูที่ไฟล์ post\_detail\_page.dart

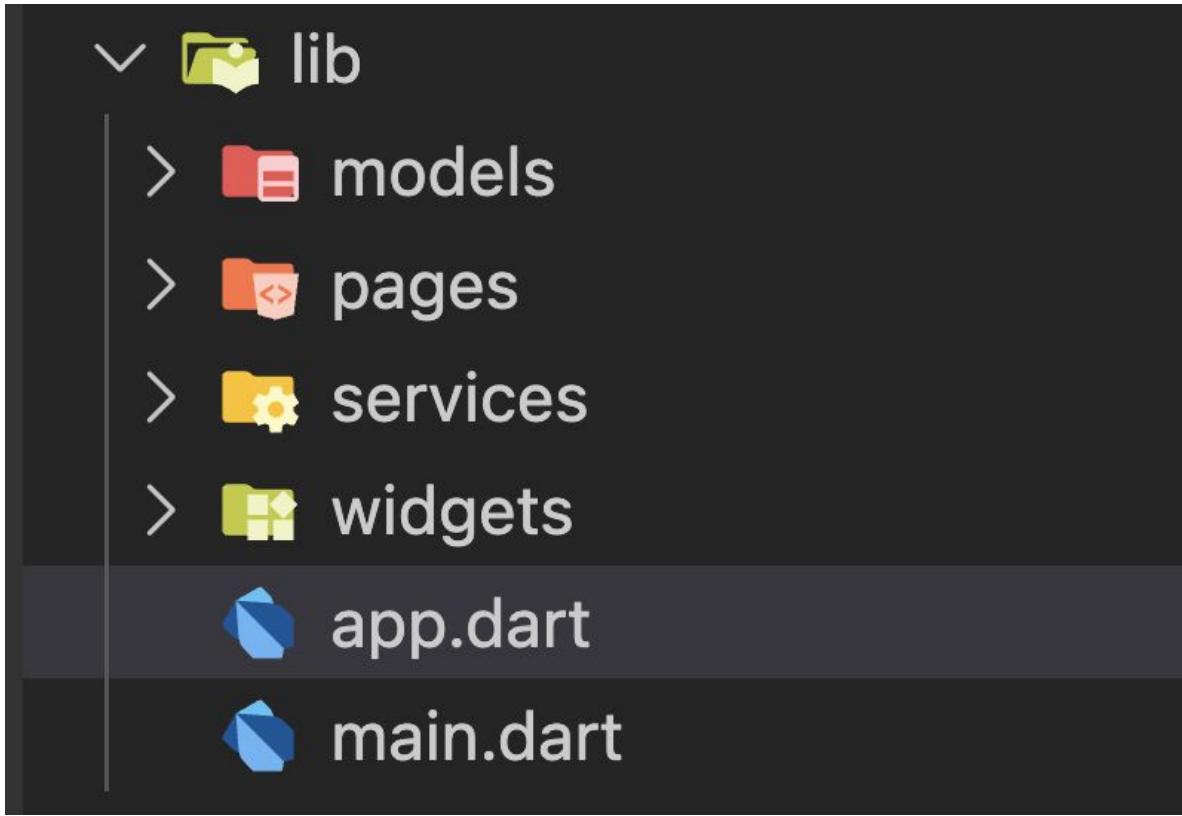
```
1 import 'package:flutter/material.dart';
2
3 import '../models/post.dart';
4 import '../services/post_api.dart';
5 import '../widgets/app_error.dart';
6 import '../widgets/app_loading.dart';
7
8 class PostDetailPage extends StatefulWidget {
9   final int postId;
10  const PostDetailPage({super.key, required this.postId});
11
12  @override
13  State<PostDetailPage> createState() => _PostDetailPageState();
14 }
15
16 class _PostDetailPageState extends State<PostDetailPage> {
17   late Future<Post> _futureDetail;
18
19   @override
20   void initState() {
21     super.initState();
22     _futureDetail = PostApi.fetchPostDetail(widget.postId);
23   }
}
```

lib > pages >  post\_detail\_page.dart > ...

```
16   class _PostDetailPageState extends State<PostDetailPage> {  
24  
25     @override  
26     Widget build(BuildContext context) {  
27       return Scaffold(  
28         appBar: AppBar(title: Text('Post #${widget.postId}')),  
29         body: FutureBuilder<Post>(  
30           future: _futureDetail,  
31           builder: (context, snapshot) {  
32             if (snapshot.connectionState == ConnectionState.waiting) {  
33               return const AppLoading();  
34             }  
35  
36             if (snapshot.hasError) {  
37               return AppError(error: snapshot.error!);  
38             }  
39           }  
40         );  
41       }  
42     }  
43   }  
44 }
```

```
16   class _PostDetailPageState extends State<PostDetailPage> {
26       Widget build(BuildContext context) {
39
40           final post = snapshot.data!;
41           return Padding(
42               padding: const EdgeInsets.all(16),
43               child: Column(
44                   crossAxisAlignment: CrossAxisAlignment.start,
45                   children: [
46                       Text(post.title, style: Theme.of(context).textTheme.titleLarge),
47                       const SizedBox(height: 12),
48                       Text(post.body),
49                   ],
50               ), // Column
51           ); // Padding
52       },
53   ), // FutureBuilder
54 ); // Scaffold
55 }
56 }
```

ประกอบແອປໃຫ້ເຮືອນຮ້ອຍ (app.dart + main.dart)



สร้างไฟล์ app.dart

```
lib > app.dart > ...
1 import 'package:flutter/material.dart';
2 import 'pages/post_list_page.dart';
3
4 class App extends StatelessWidget {
5     const App({super.key});
6
7     @override
8     Widget build(BuildContext context) {
9         return MaterialApp(
10             title: 'Post Reader',
11             theme: ThemeData(useMaterial3: true),
12             home: const PostListPage(),
13         ); // MaterialApp
14     }
15 }
```

สร้างไฟล์ app.dart

lib >  main.dart > ...

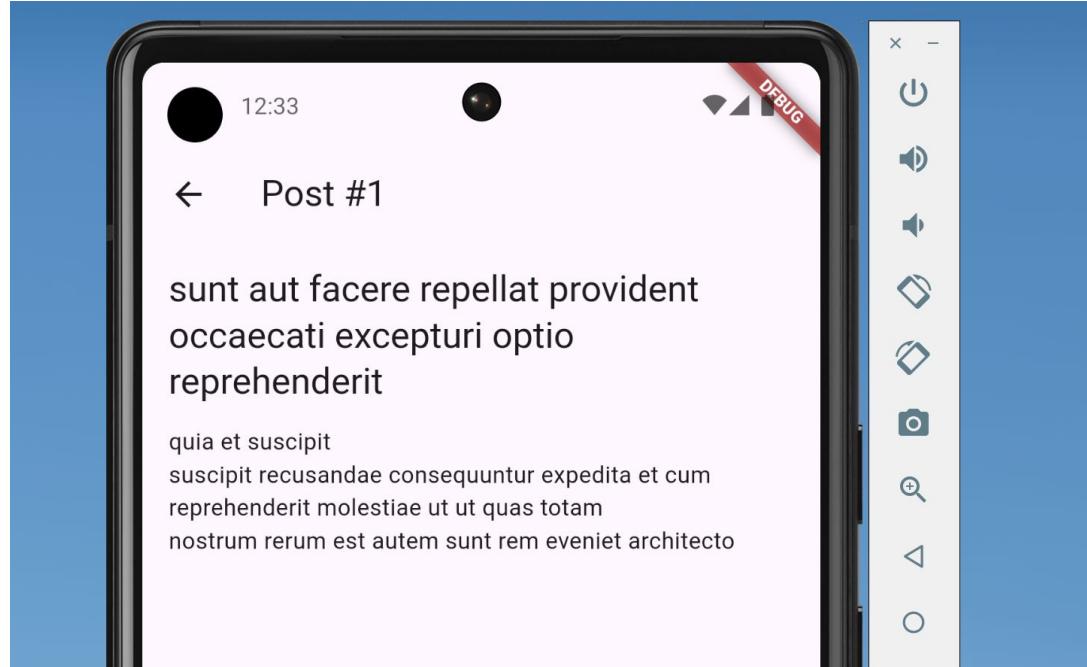
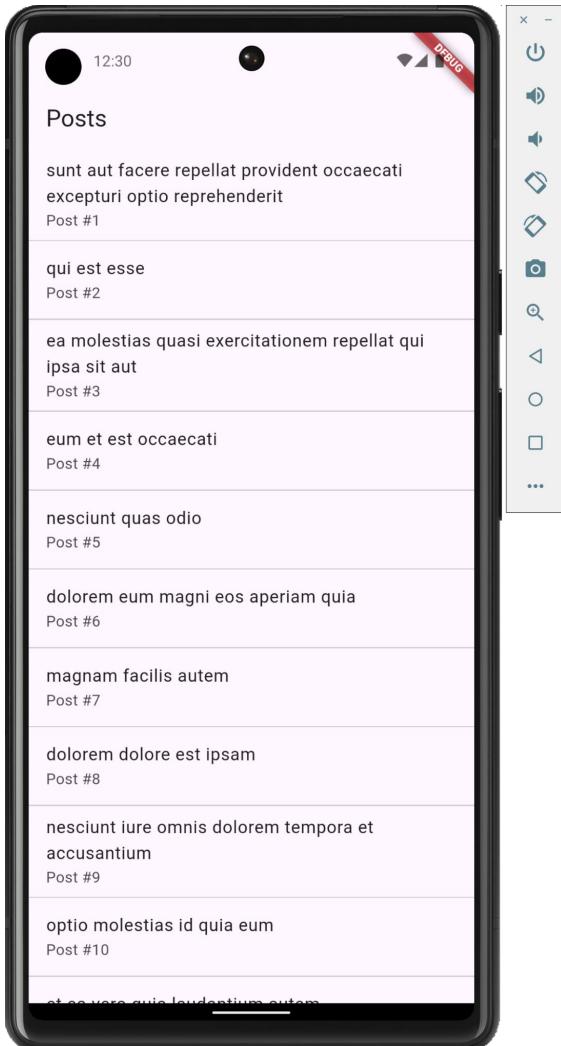
```
1 import 'package:flutter/material.dart';
2 import 'app.dart';
```

3

Run | Debug | Profile

```
4 void main() => runApp(const App());
5 |
```

แก้ไขในไฟล์ lib/main.dart



ทดสอบได้เลย !

**Get started**[Set up Flutter](#)[Install Flutter](#)[Learn Flutter](#)[Stay up to date](#)[App solutions](#)[AI solutions](#)**User interface**[Introduction](#)[Widget catalog](#)[Layout](#)[Adaptive & responsive design](#)[Design & theming](#)[Interactivity](#)[Assets & media](#)[Navigation & routing](#)[Animations & transitions](#)[Accessibility](#)[Internationalization](#)**Beyond UI**[Data & backend](#)[App architecture](#)[Platform integration](#)[Packages & plugins](#)

Even if you missed it, you can watch the recording of the December 17th livestream and Q&amp;A!

[Cookbook](#) > Networking

# Flutter networking cookbook

A catalog of recipes for networking in your Flutter app.

## Communicate with WebSockets

How to connect to a web socket.

## Parse JSON in the background

How to perform a task in the background.

## Delete data on the internet

How to use the http package to delete data on the internet.

## Update data over the internet

How to use the http package to update data over the internet.

## Send data to the internet

How to use the http package to send data over the internet.