
MOBILE PROGRAMMING
WITH FLUTTER - DAY6
เริ่มพัฒนาแอปพลิเคชันบนมือถือด้วย Flutter
API
REST
JSON
HTTP
Bant
Next
Content

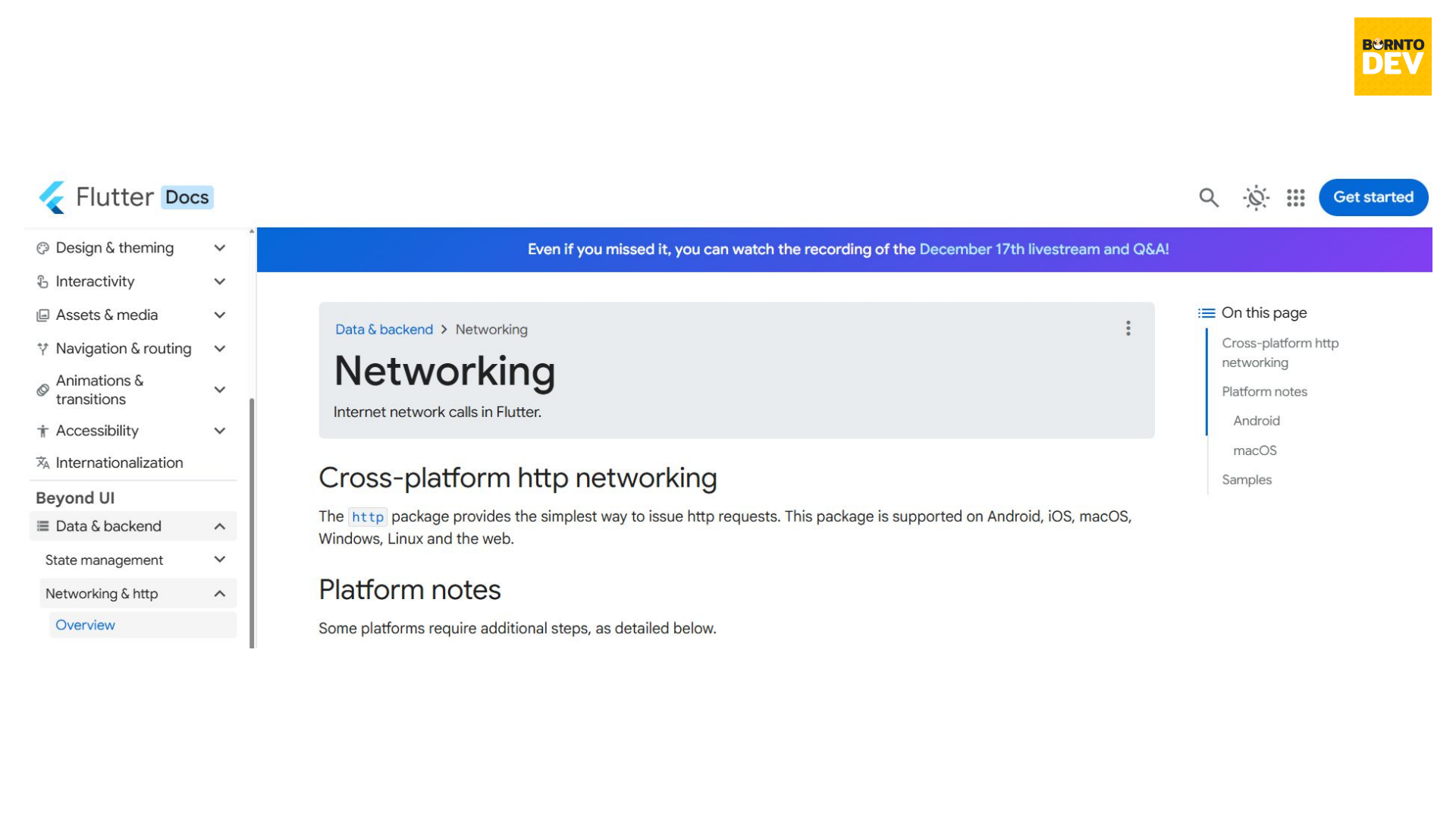
การบ้านสำหรับเช็คชื่อรอบนี้

ตรวจรายละเอียดแผนการจัดทำ

PROJECT รวมกลุ่ม 3 คน

เรากำลังทำอะไรใน “API 101”


UI (หน้าแอป)
เรียก Service
ส่ง HTTP Request
ได้ JSON
แปลงเป็น Model


Flutter Docs
Get started
Design & theming
Interactivity
Assets & media
Navigation & routing
Animations & transitions
Accessibility
Internationalization
Beyond UI
Data & backend
State management
Networking & http
Overview
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.

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

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.

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

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.

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

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.

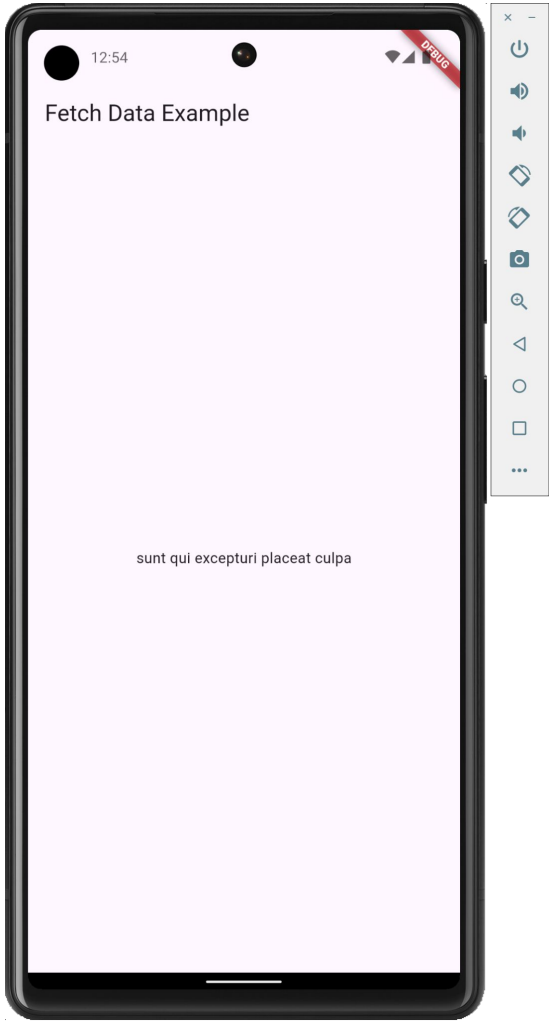
Flutter calls the `build()` method every time it needs to change anything in the view, and this happens surprisingly often. The `fetchAlbum()` method, if placed inside `build()`, is repeatedly called on each rebuild causing the app to slow down.

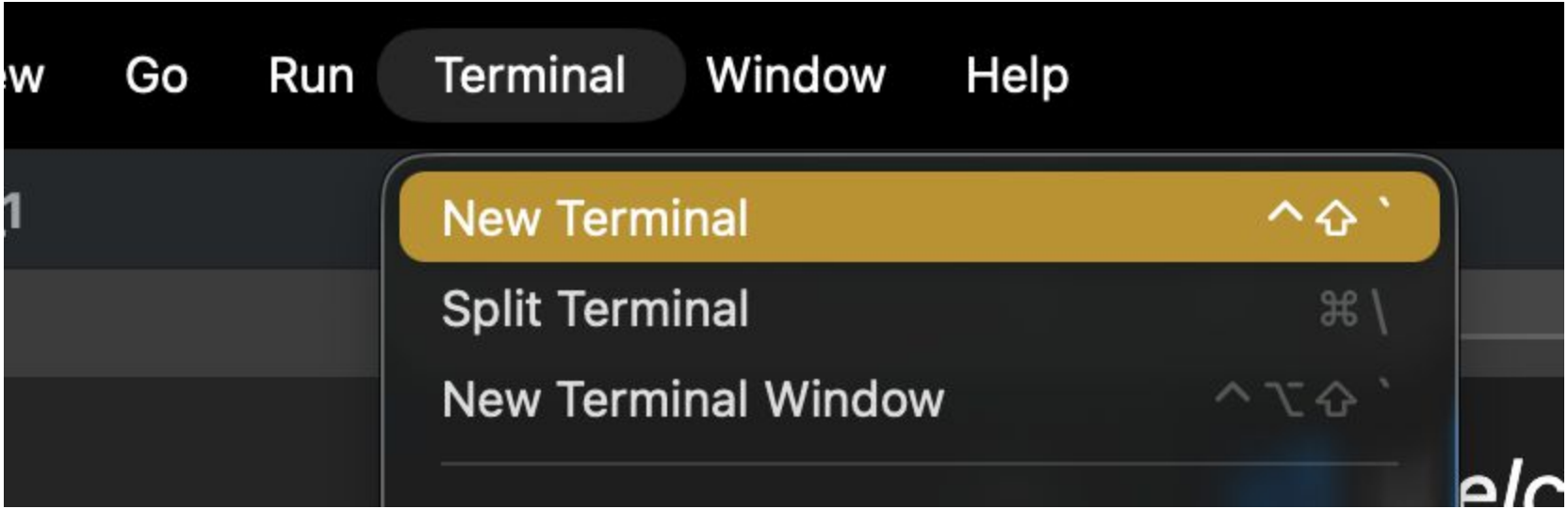
Storing the `fetchAlbum()` result in a state variable ensures that the `Future` is executed only once and then cached for subsequent rebuilds.

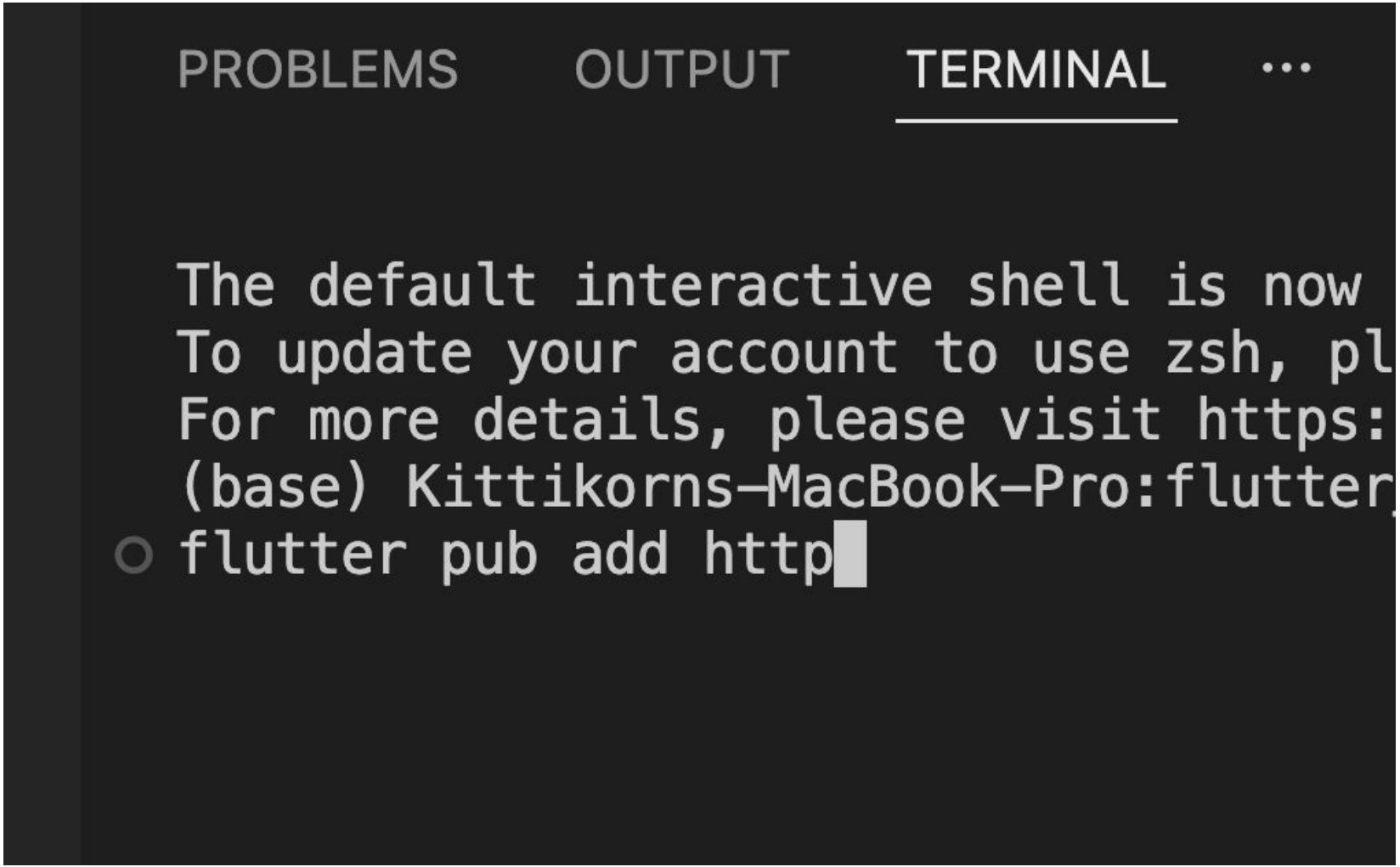
Testing

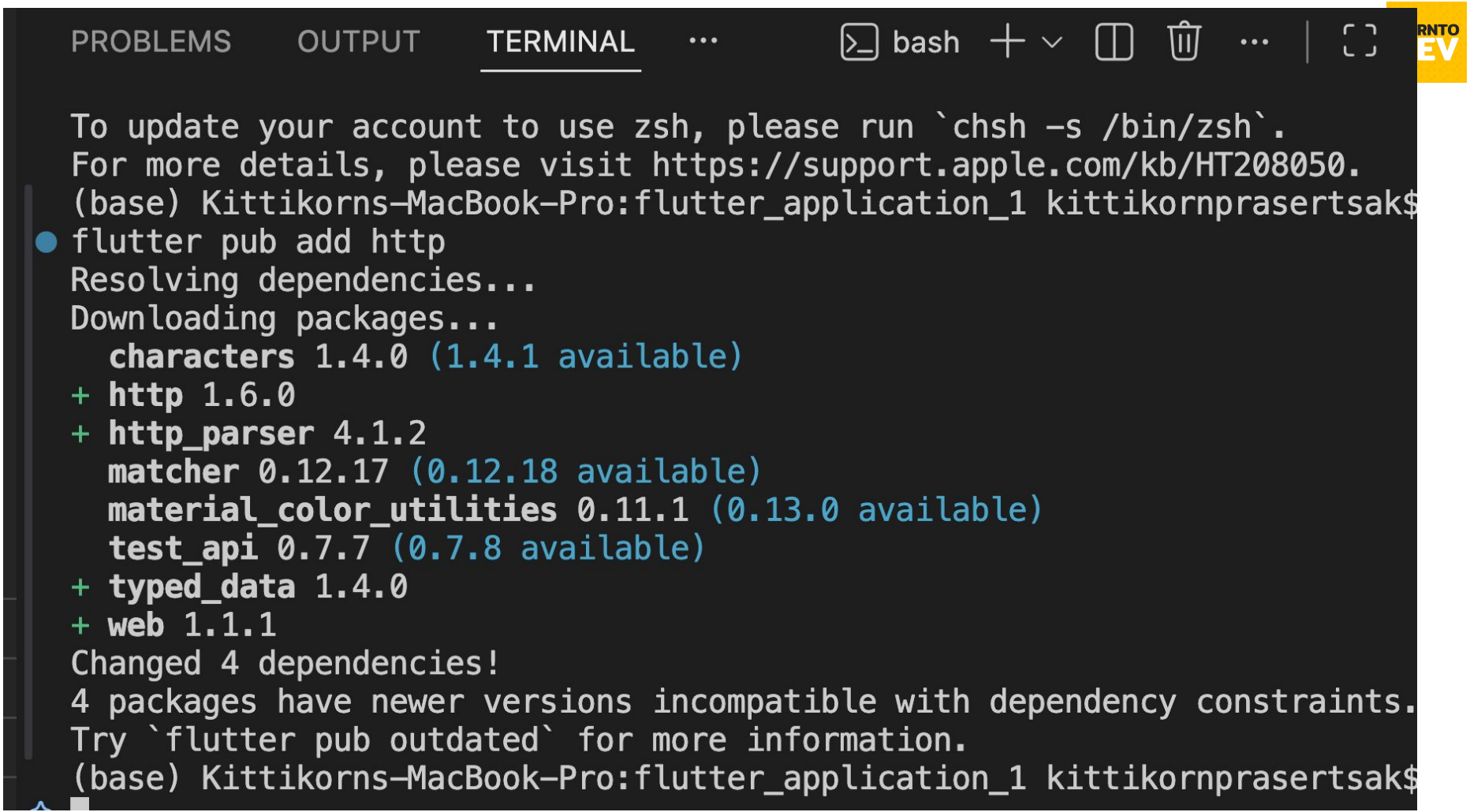
For information on how to test this functionality, see the following recipes:

- Introduction to unit testing
- Mock dependencies using Mockito


12:54
DEBUG
Fetch Data Example
sunt qui excepturi placeat culpa


Go
Run
Terminal
Window
Help
New Terminal
Split Terminal
New Terminal Window


PROBLEMS
OUTPUT
TERMINAL
The default interactive shell is now
To update your account to use zsh, pl
For more details, please visit https:
(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$

main.dart

lib > main.dart > ...

```dart
import 'dart:async';
import 'dart:convert';

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

Future<Album> fetchAlbum() async {
  final response = await http.get(
    Uri.parse('https://jsonplaceholder.typicode.com/albums/2'),
  );

  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');
  }
}
```

main.dart

lib > main.dart > ...

```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.'),
    };
  }
}

```

main.dart

lib > main.dart > ...

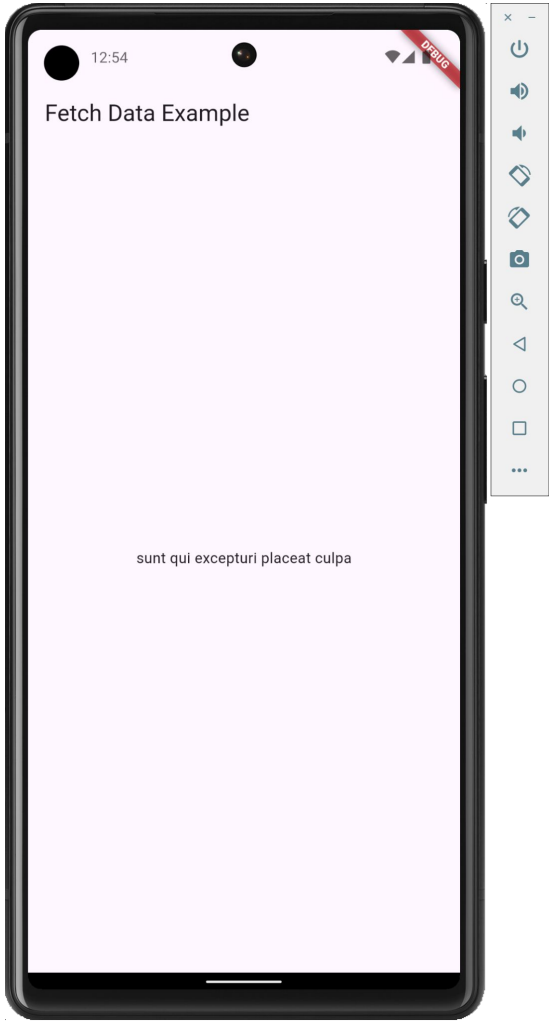
Run | Debug | Profile

```dart
void main() => runApp(const MyApp());

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

  @override
  State<MyApp> createState() => _MyAppState();
}

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

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

main.dart ●

lib > main.dart > _MyAppState

```dart
class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Fetch Data Example',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
      ), // ThemeData
      home: Scaffold(
        appBar: AppBar(title: const Text('Fetch Data Example')),
        body: Center(
          child: 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();
            },
          ), // FutureBuilder
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```


12:54
DEBUG
Fetch Data Example
sunt qui excepturi placeat culpa

ตัวอย่างการทำ Authen Request

```dart
Future<Album> fetchAlbum() async {
  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'},
  );
  final responseJson = jsonDecode(response.body) as Map<String, dynamic>;

  return Album.fromJson(responseJson);
}
```

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

Make authenticated requests

How to fetch authorized data from a web service.

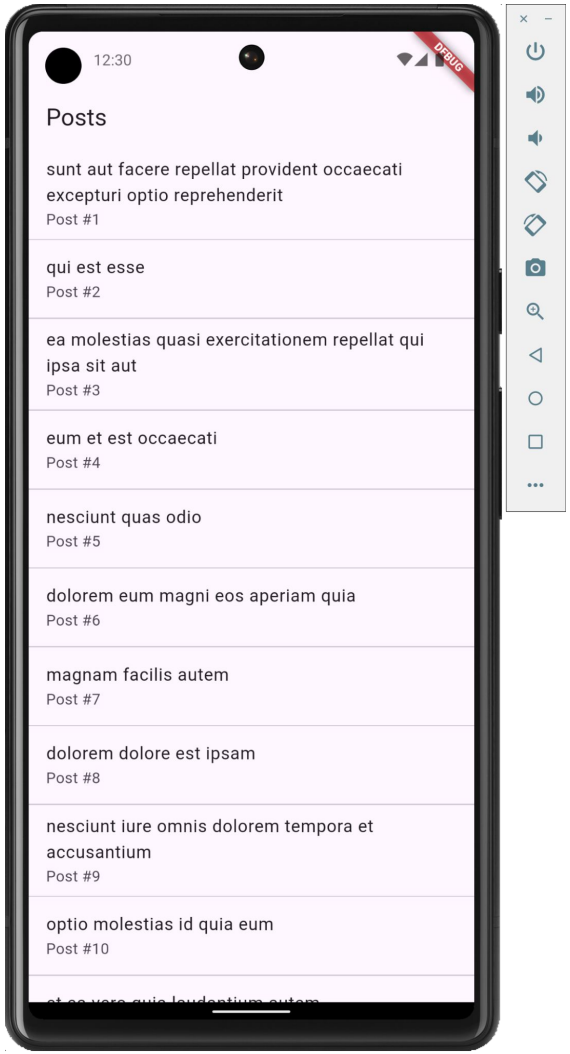
To fetch data from most web services, you need to provide authorization. There are many ways to do this, but perhaps the most common uses the `Authorization` HTTP header.

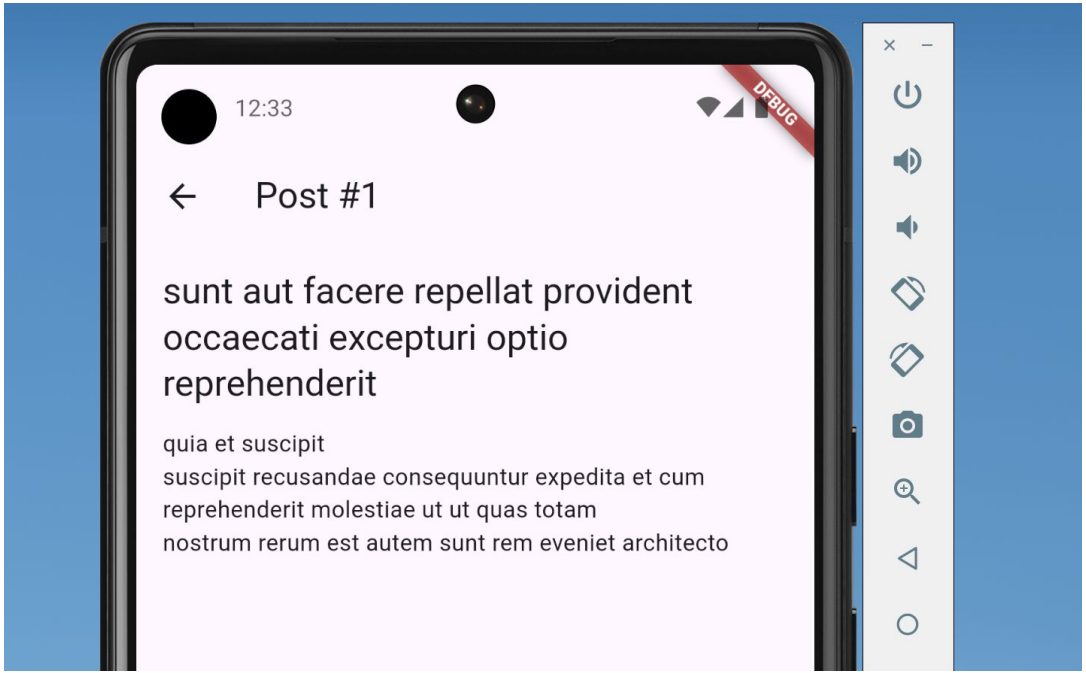
Add authorization headers

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

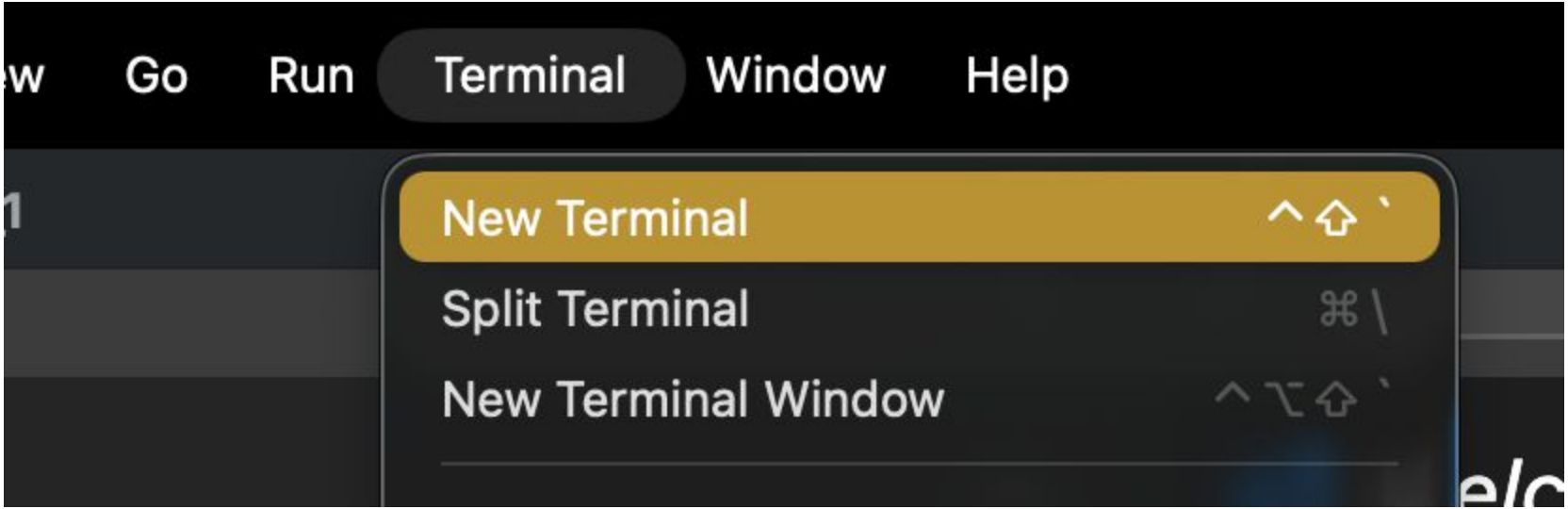
```dart
final response = await http.get(
  Uri.parse('https://jsonplaceholder.typicode.com/albums/1'),
  // Send authorization headers to the backend.
  headers: {HttpHeaders.authorizationHeader: 'Basic your_api_token_here'},
);
```

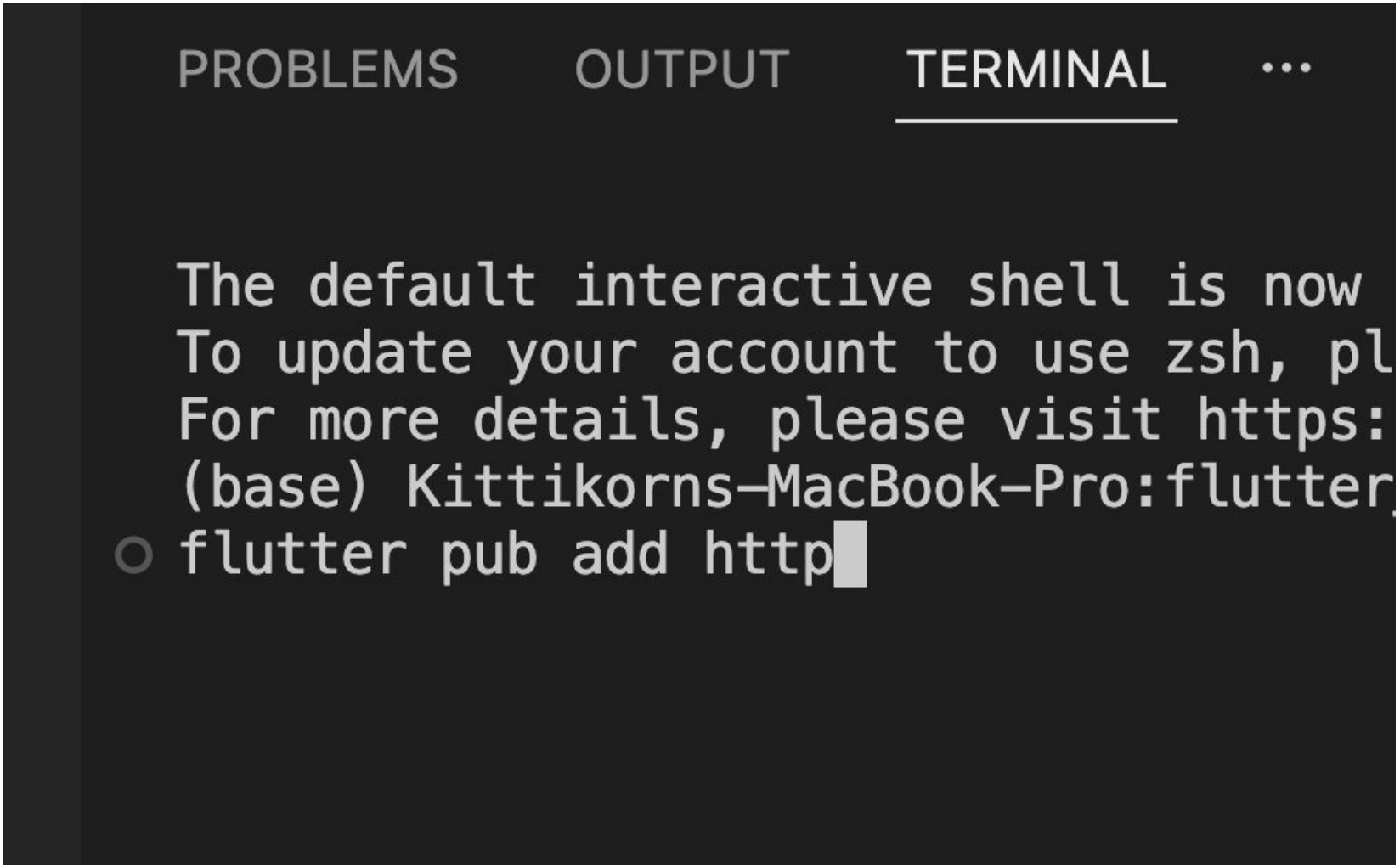
ลองดูตัวอย่างของตัวเองกัน

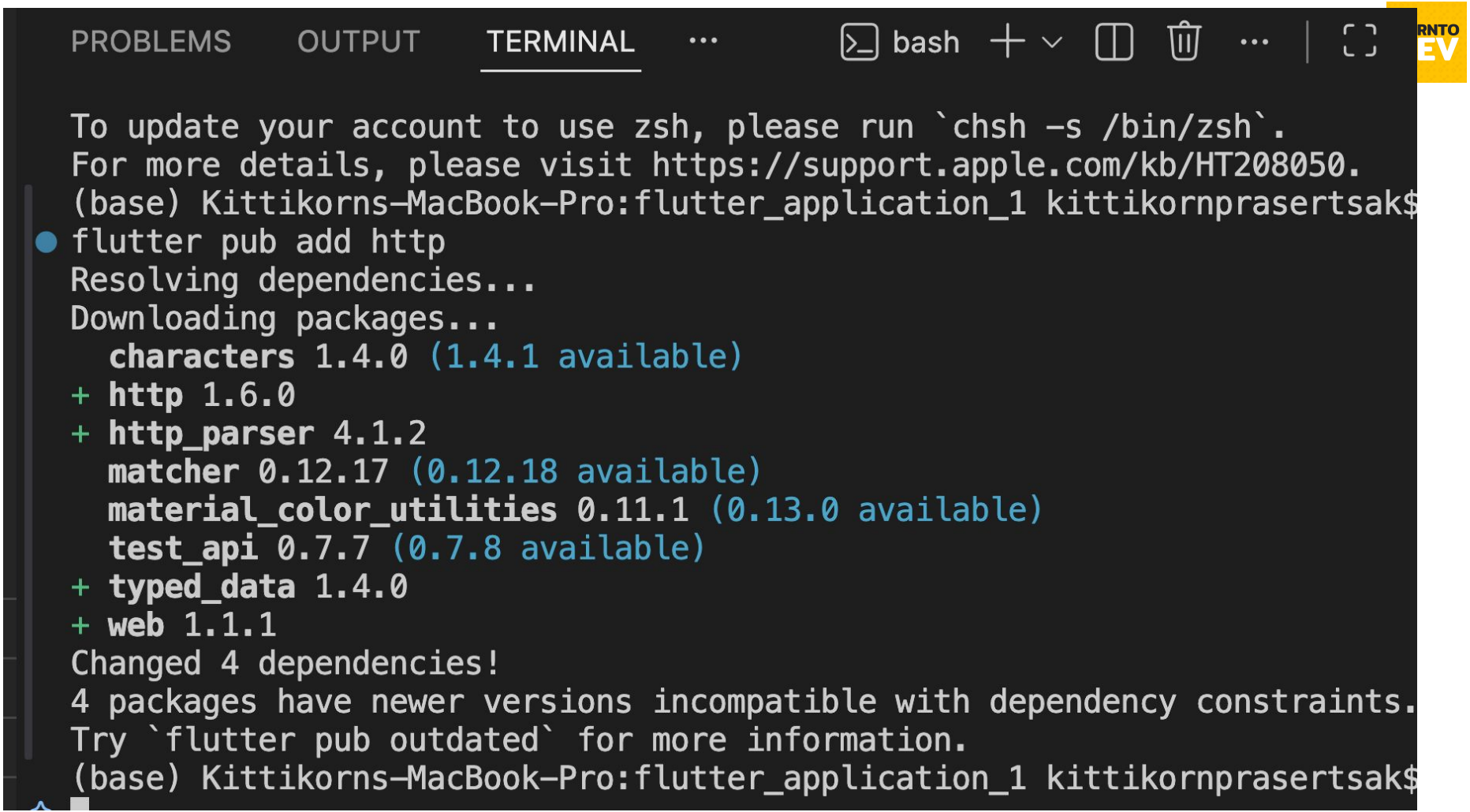


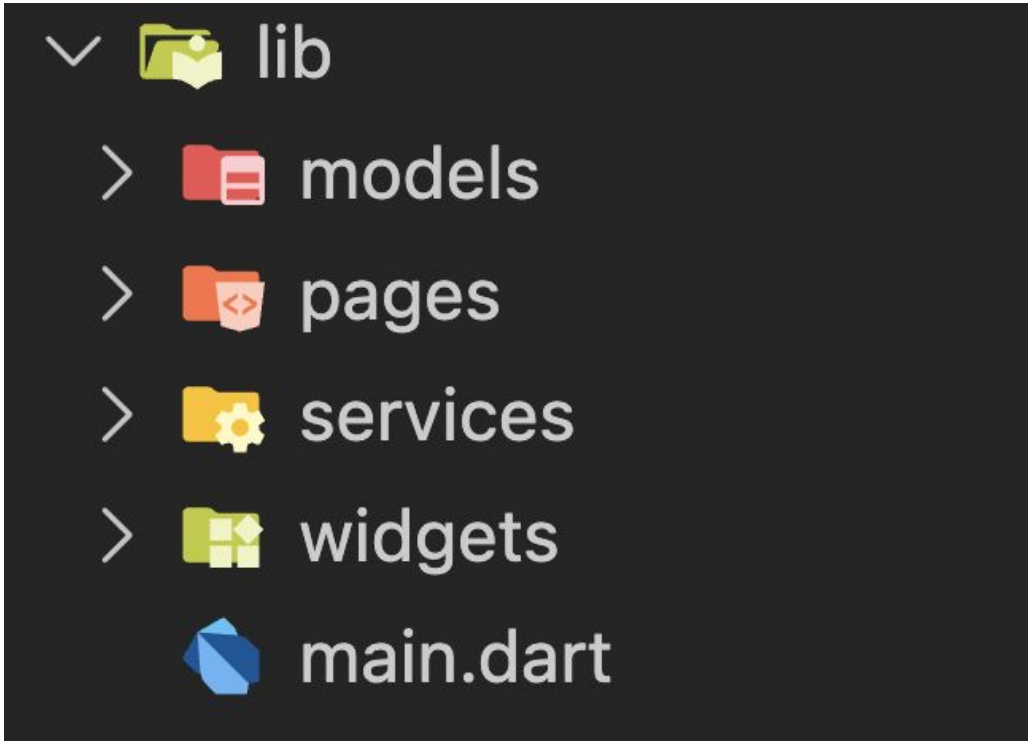


ปลายทางอยากได้แบบนี้

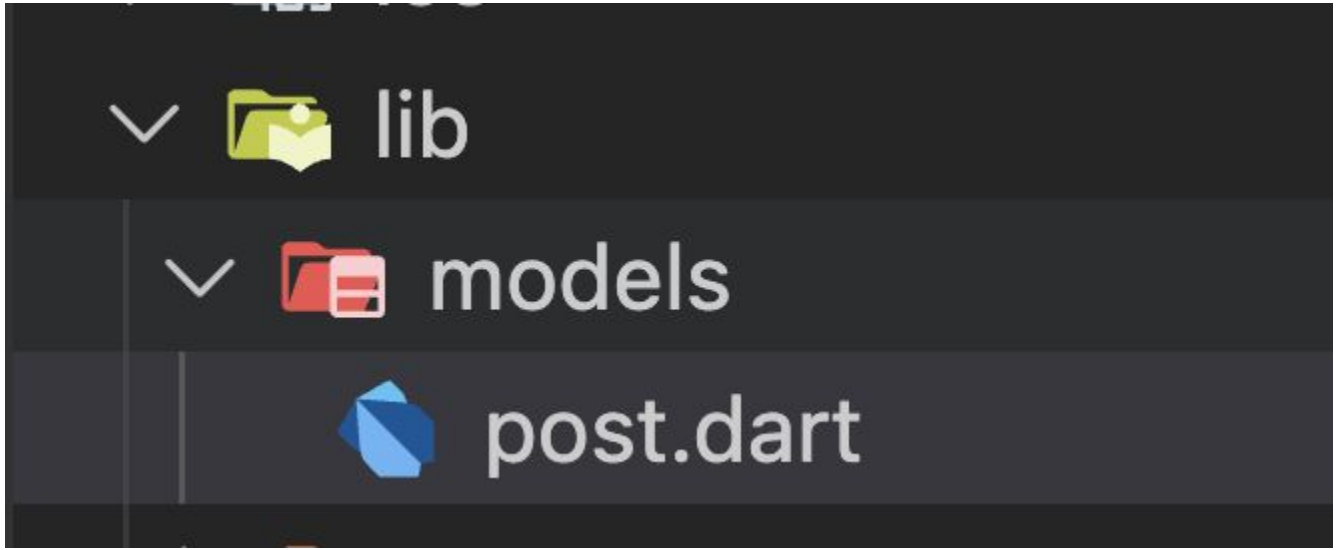

Go
Run
Terminal
Window
Help
New Terminal
Split Terminal
New Terminal Window


PROBLEMS
OUTPUT
TERMINAL
The default interactive shell is now
To update your account to use zsh, pl
For more details, please visit https:
(base) Kittikorns-MacBook-Pro:flutter
flutter pub add http

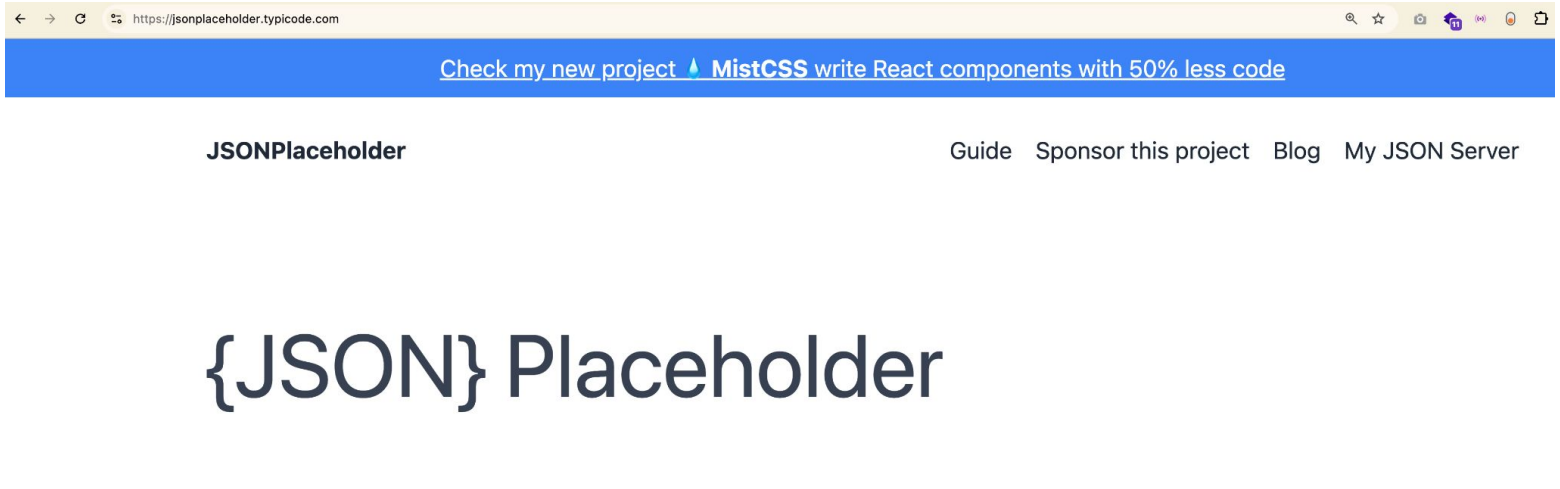

PROBLEMS
OUTPUT
TERMINAL
bash
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Kittikorns-MacBook-Pro:flutter_application_1 kittikornprasertsak$
flutter pub add http
Resolving dependencies...
Downloading packages...
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



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\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem 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 doloremque 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\nvoluptatem 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\nvoluptate dolores velit et doloremque molestiae"
  },
  {
    "userId": 1,
    "id": 7,
    "title": "magnam 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"
  },
```

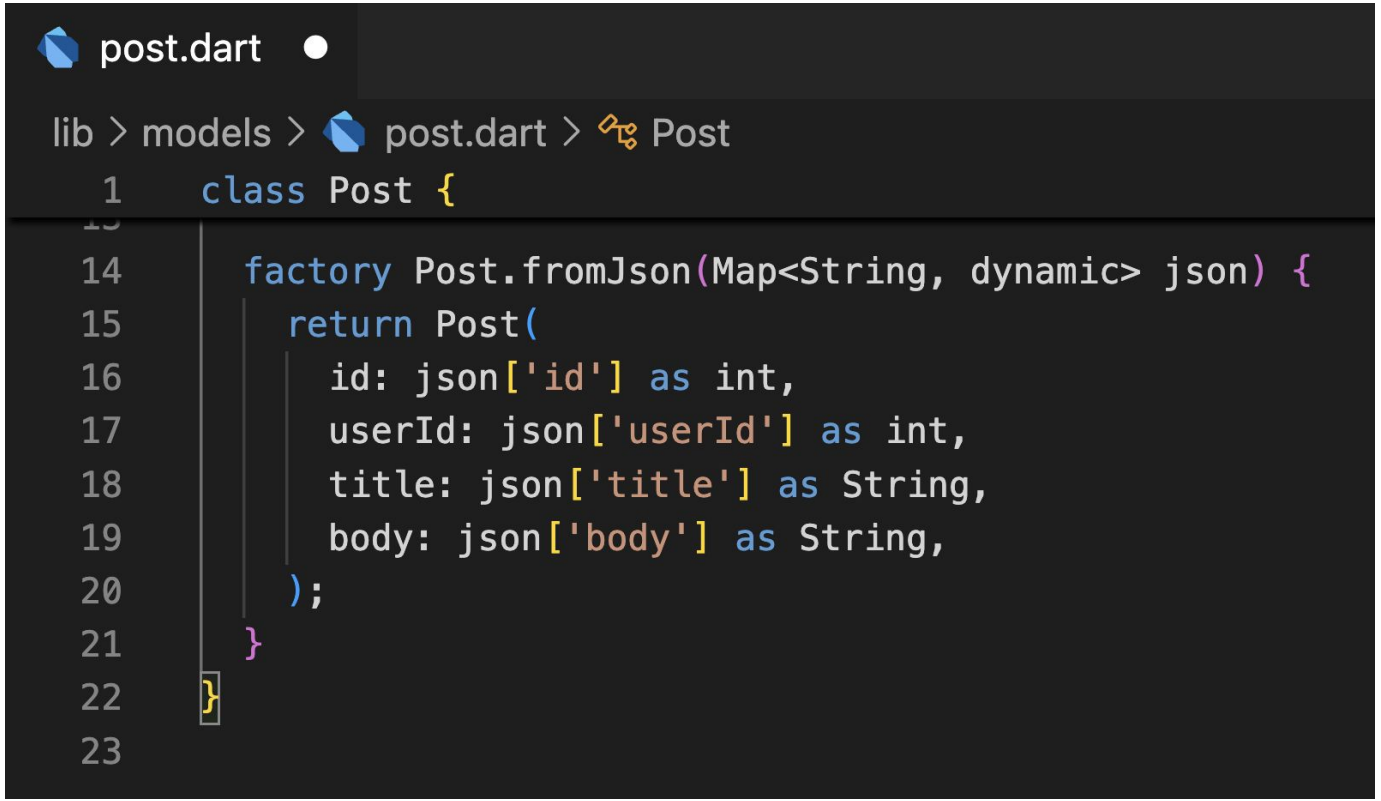

https://jsonplaceholder.typicode.com/posts

pubspec.yaml post.dart

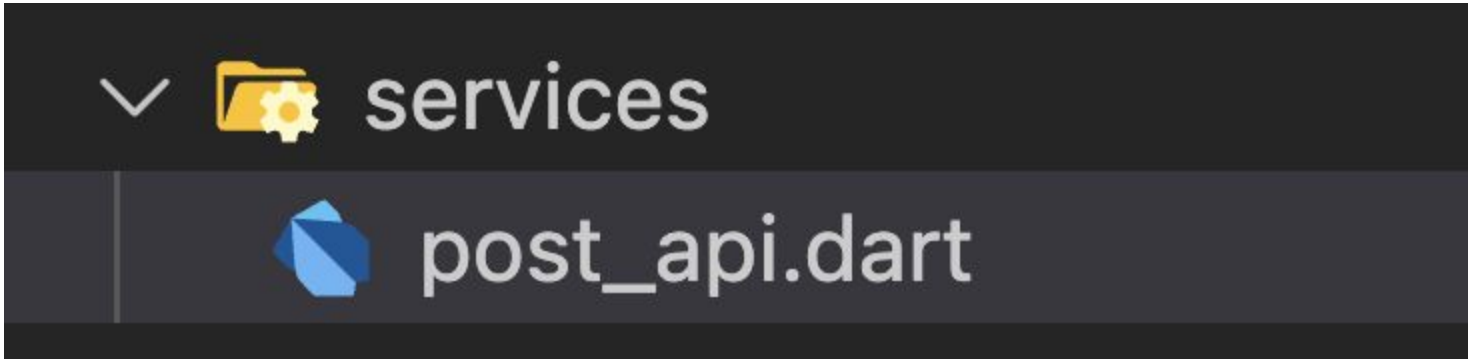
lib > models > post.dart > ...

```dart
class Post {
  final int id;
  final int userId;
  final String title;
  final String body;

  Post({
    required this.id,
    required this.userId,
    required this.title,
    required this.body,
  });
}
```

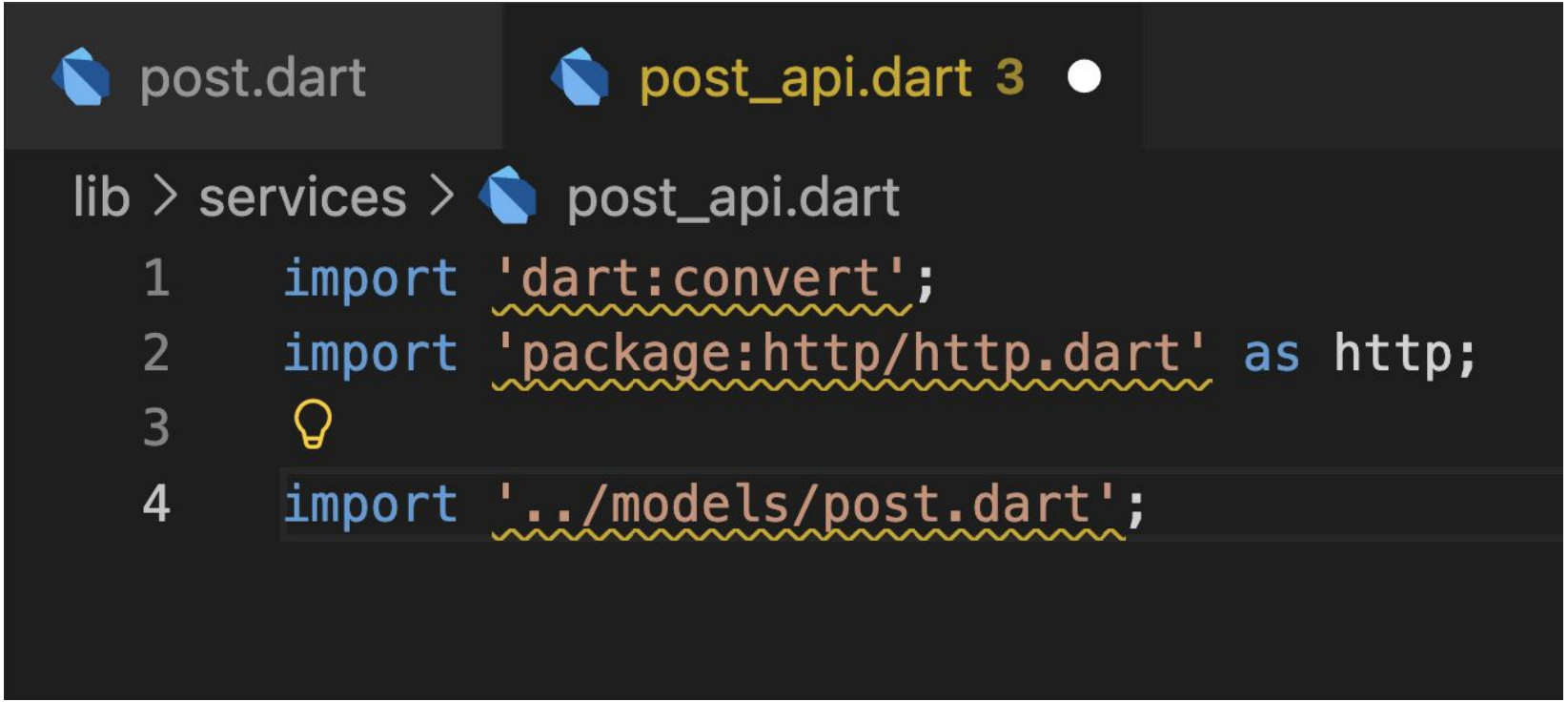
สร้าง Model ใน post.dart ให้เรียบร้อย



ทำการ Decode ออกมาเป็น Map / List เพื่อจัดการต่อได้



สร้างไฟล์ post_api.dart



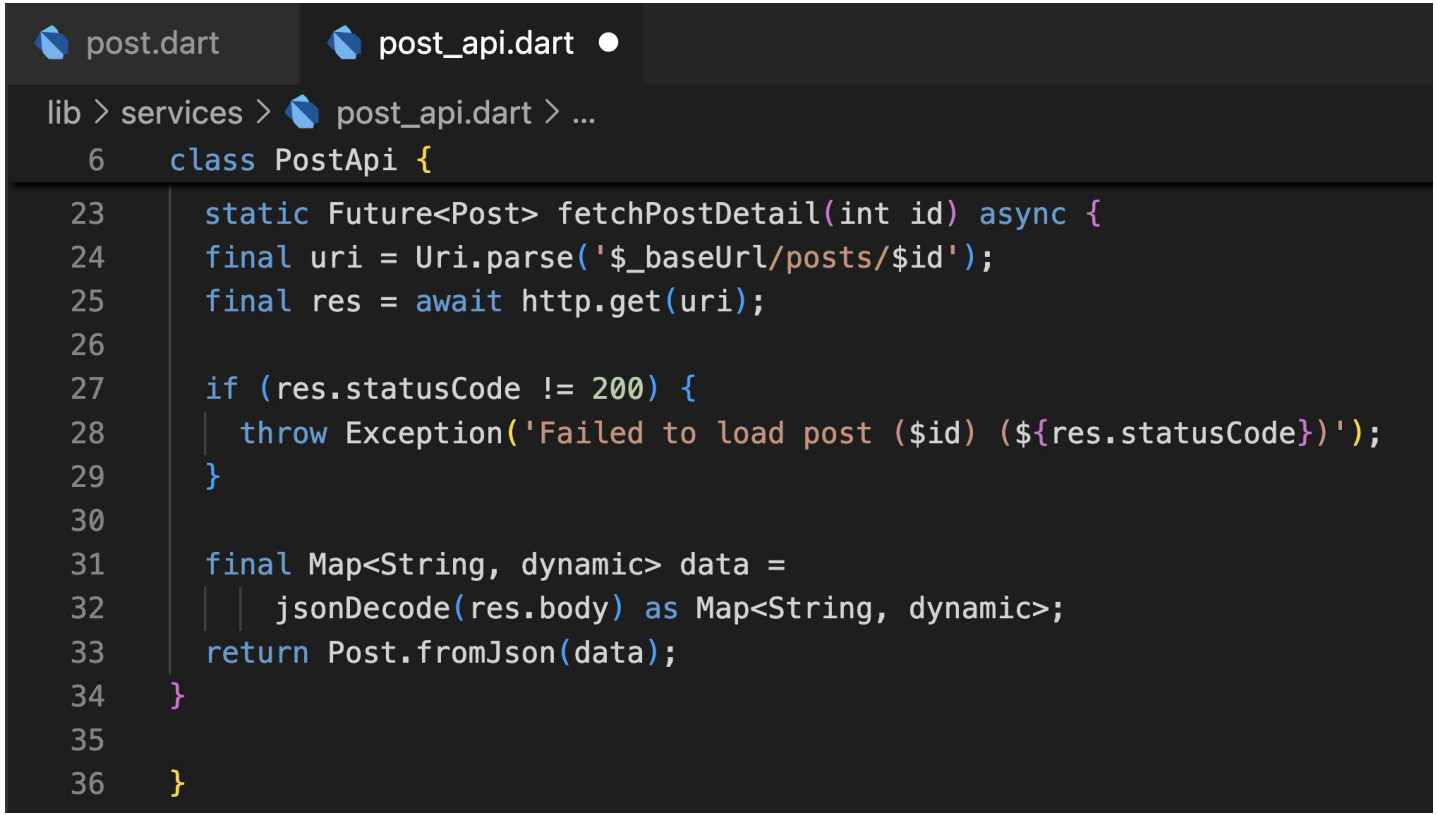
เริ่มจาก import ที่จำเป็น

post.dart | post_api.dart ●

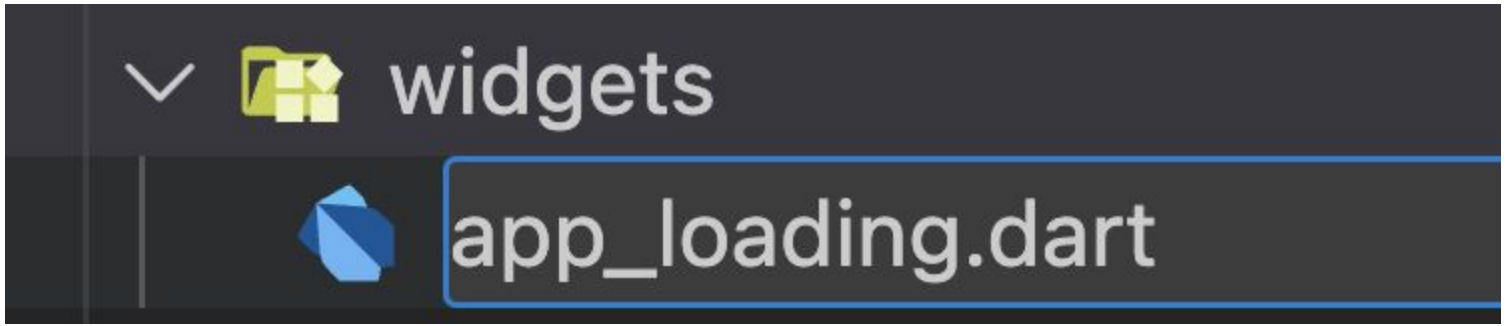
lib > services > post_api.dart > ...

```dart
class PostApi {
  static const _baseUrl = 'https://jsonplaceholder.typicode.com';

  static Future<List<Post>> fetchPosts() async {
    final uri = Uri.parse('$_baseUrl/posts');
    final res = await http.get(uri);

    if (res.statusCode != 200) {
      throw Exception('Failed to load posts (${res.statusCode})');
    }

    final List<dynamic> data = jsonDecode(res.body) as List<dynamic>;
    return data
        .map((e) => Post.fromJson(e as Map<String, dynamic>))
        .toList();
  }
}
```

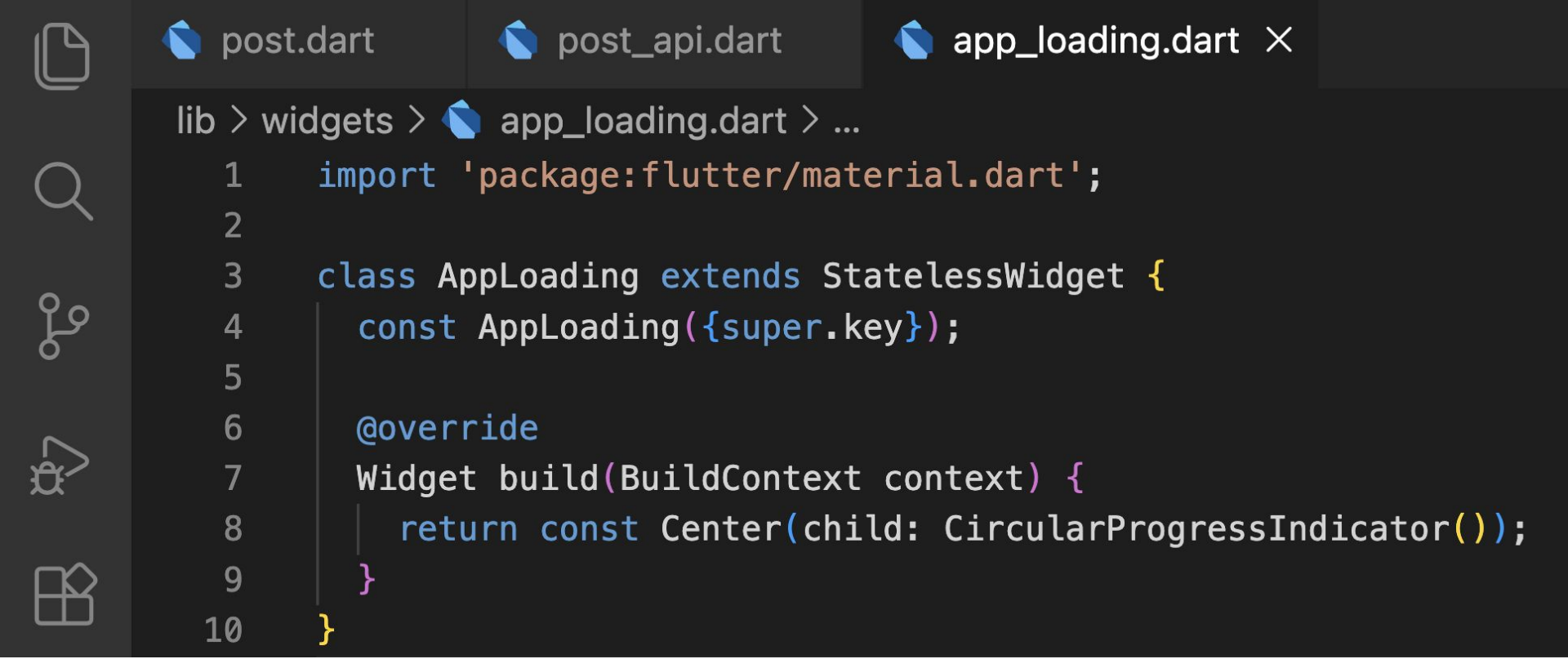
ทำเมธอดดึง “รายการโพสต์”



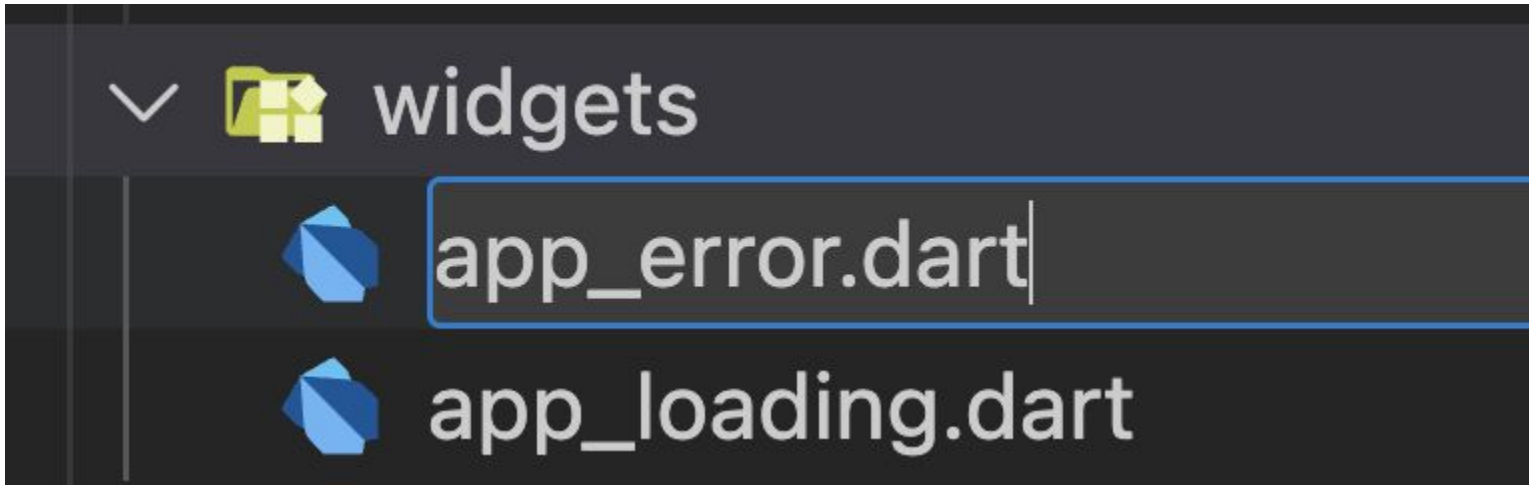
ทำเมธอดดึง “รายละเอียดโพสต์”



สร้างไฟล์ app_loading.dart ไว้ใน widgets



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 > ...

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

class AppError extends StatelessWidget {
  final Object error;
  final VoidCallback? onRetry;

  const AppError({
    super.key,
    required this.error,
    this.onRetry,
  });

  @override
  Widget build(BuildContext context) {
    return Center(
    );
  }
}

```

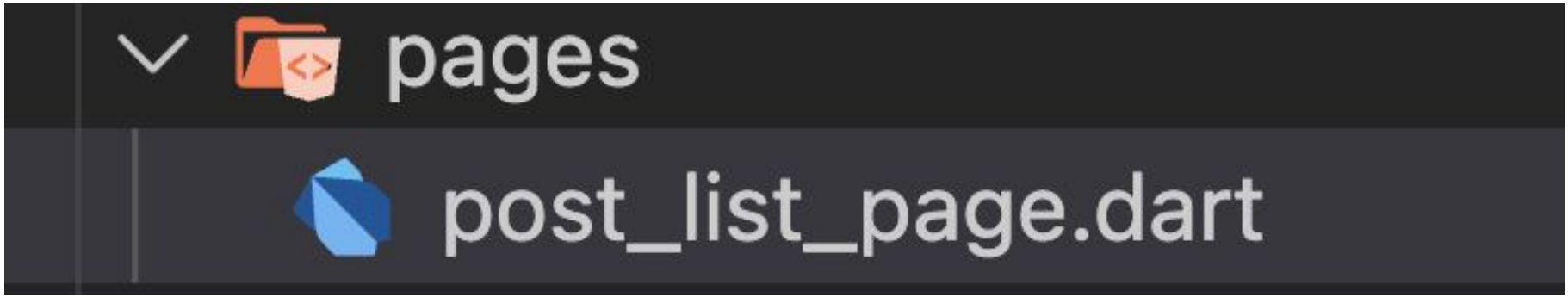
สร้างไฟล์ app_error.dart ไว้ใน widgets

post_api.dart | app_loading.dart | app_error.dart

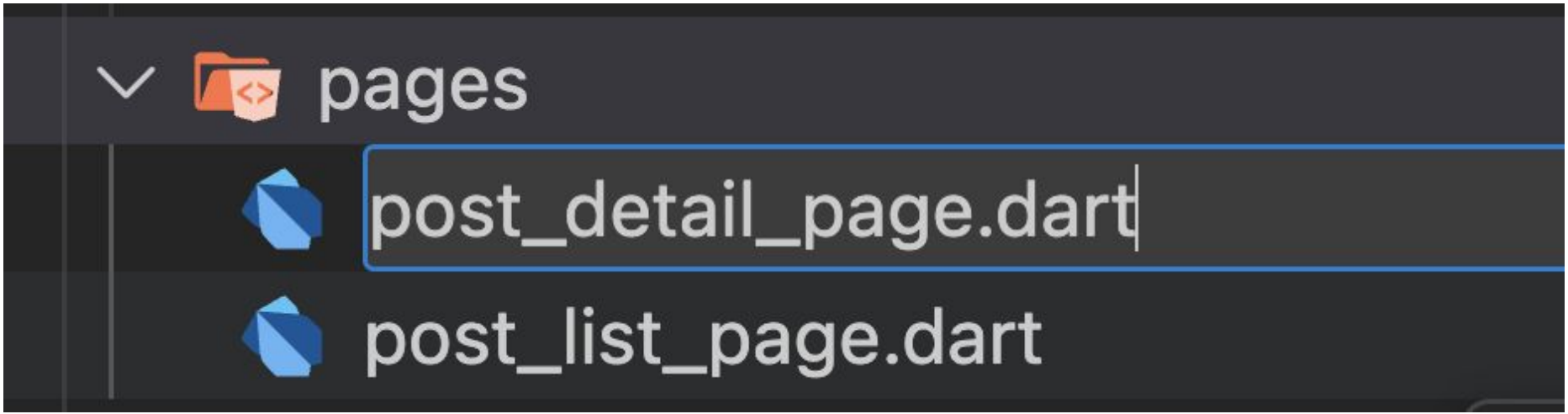
lib > widgets > app_error.dart > AppError

```dart
class AppError extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Padding(
        padding: const EdgeInsets.all(16),
        child: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            Text('❌ $error'),
            const SizedBox(height: 12),
            if (onRetry != null)
              ElevatedButton(
                onPressed: onRetry,
                child: const Text('Retry'),
              ), // ElevatedButton
          ],
        ), // Column
      ), // Padding
    ); // Center
  }
}
```

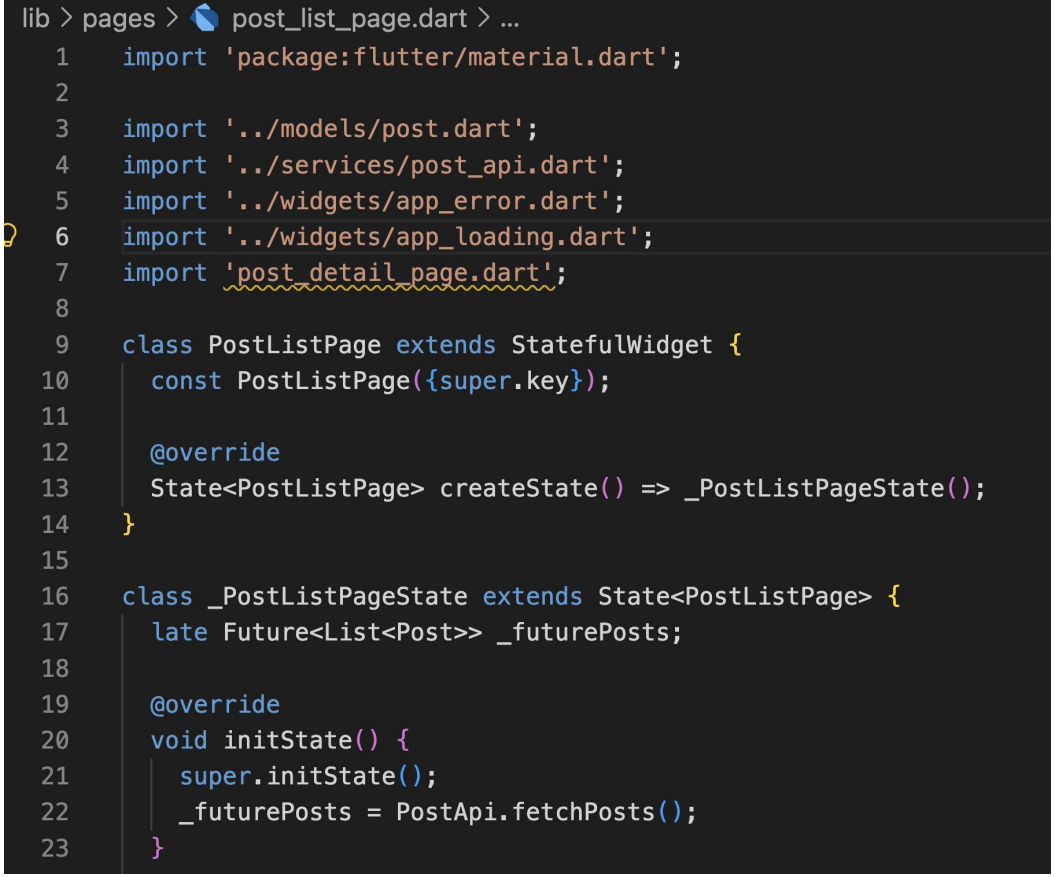
สร้างไฟล์ app_error.dart ไว้ใน widgets



สร้างไฟล์ post_list_page.dart ไว้ใน pages



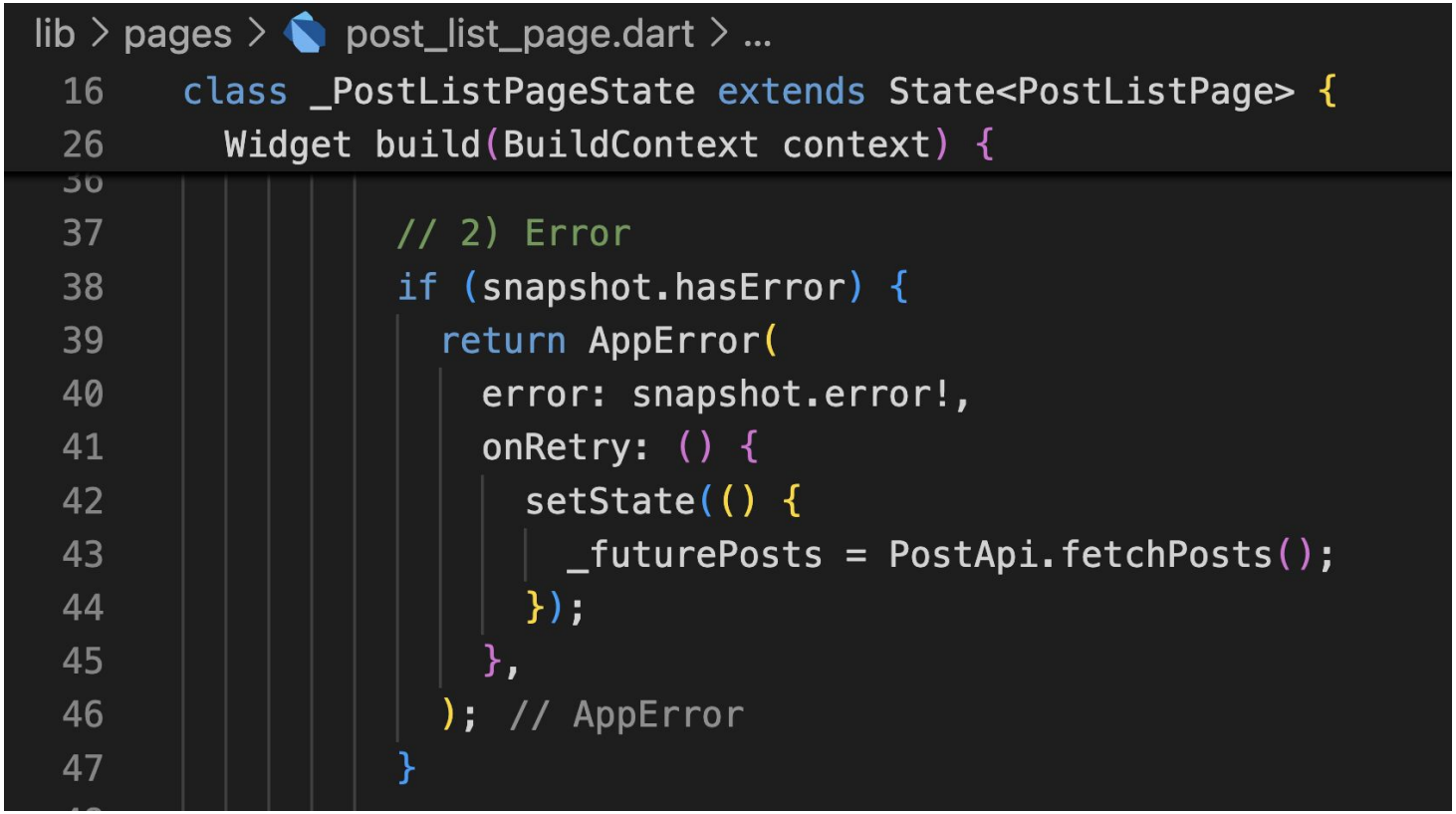
สร้างไฟล์ post_detail_page.dart ไว้ใน Pages



สร้างไฟล์ post_list_page.dart ไว้ใน pages

```
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            }
```

สร้างไฟล์ post_list_page.dart ไว้ใน pages

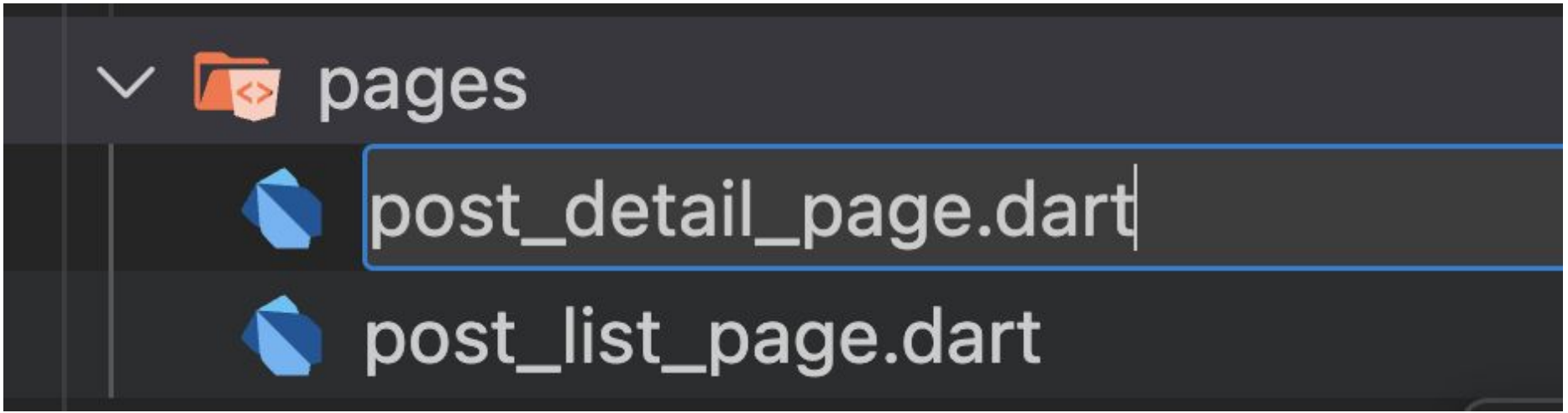


สร้างไฟล์ post_list_page.dart ไว้ใน pages

lib > pages > post_list_page.dart > ...

```dart
16  class _PostListPageState extends State<PostListPage> {
26    Widget build(BuildContext context) {
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) {
63 builder: (_) => PostDetailPage(postId: p.id),
64), // MaterialPageRoute
65);
66 },
67); // ListTile
68 },
69); // ListView.separated
70 },
71), // FutureBuilder
72); // Scaffold
73 }
74 }
75



มาดูที่ไฟล์ post_detail_page.dart

lib > pages > post_detail_page.dart > ...

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

import '../models/post.dart';
import '../services/post_api.dart';
import '../widgets/app_error.dart';
import '../widgets/app_loading.dart';

class PostDetailPage extends StatefulWidget {
  final int postId;
  const PostDetailPage({super.key, required this.postId});

  @override
  State<PostDetailPage> createState() => _PostDetailPageState();
}

class _PostDetailPageState extends State<PostDetailPage> {
  late Future<Post> _futureDetail;

  @override
  void initState() {
    super.initState();
    _futureDetail = PostApi.fetchPostDetail(widget.postId);
  }
```

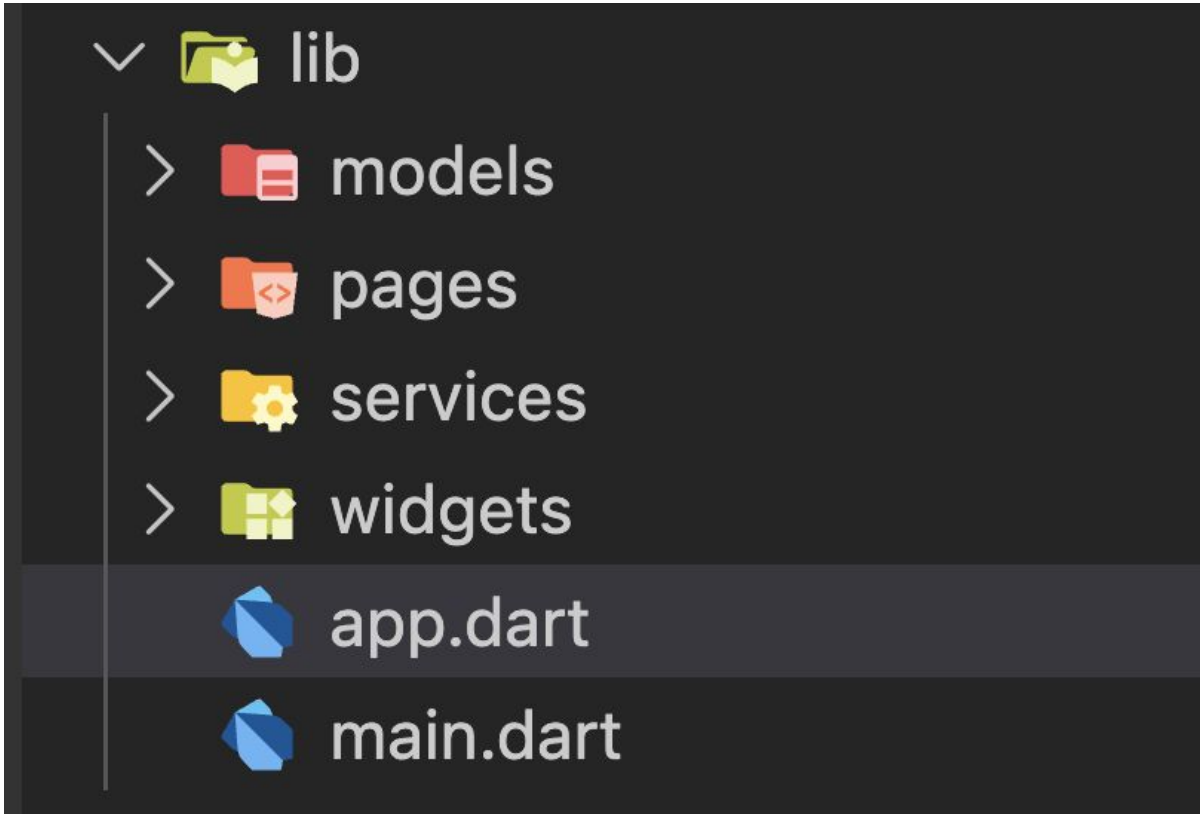
BORNTO DEV

```dart
class _PostDetailPageState extends State<PostDetailPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Post #${widget.postId}')),
      body: FutureBuilder<Post>(
        future: _futureDetail,
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const AppLoading();
          }

          if (snapshot.hasError) {
            return AppError(error: snapshot.error!);
          }
```

lib > pages > post_detail_page.dart > ...

```
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  }
```

ประกอบแอปให้เรียบร้อย (app.dart + main.dart)

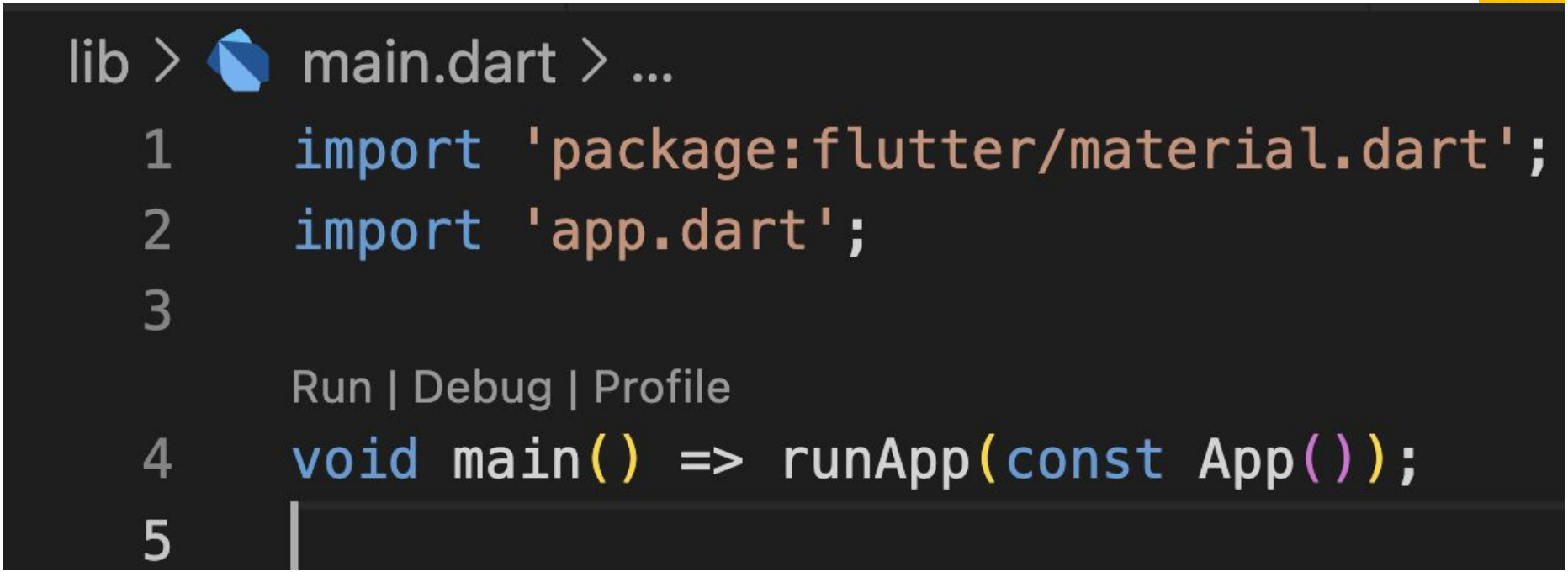


สร้างไฟล์ app.dart

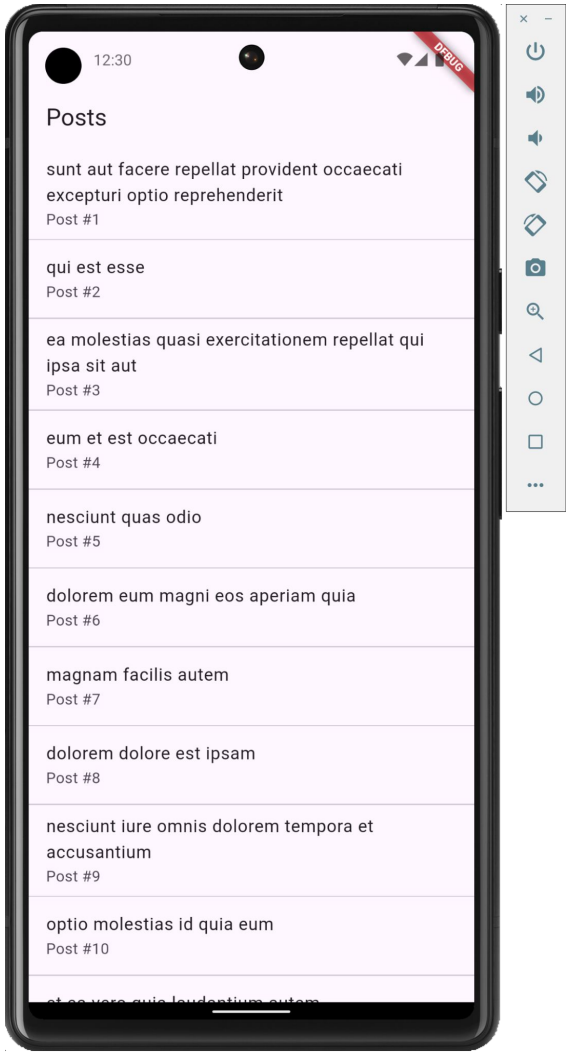
lib > app.dart > ...

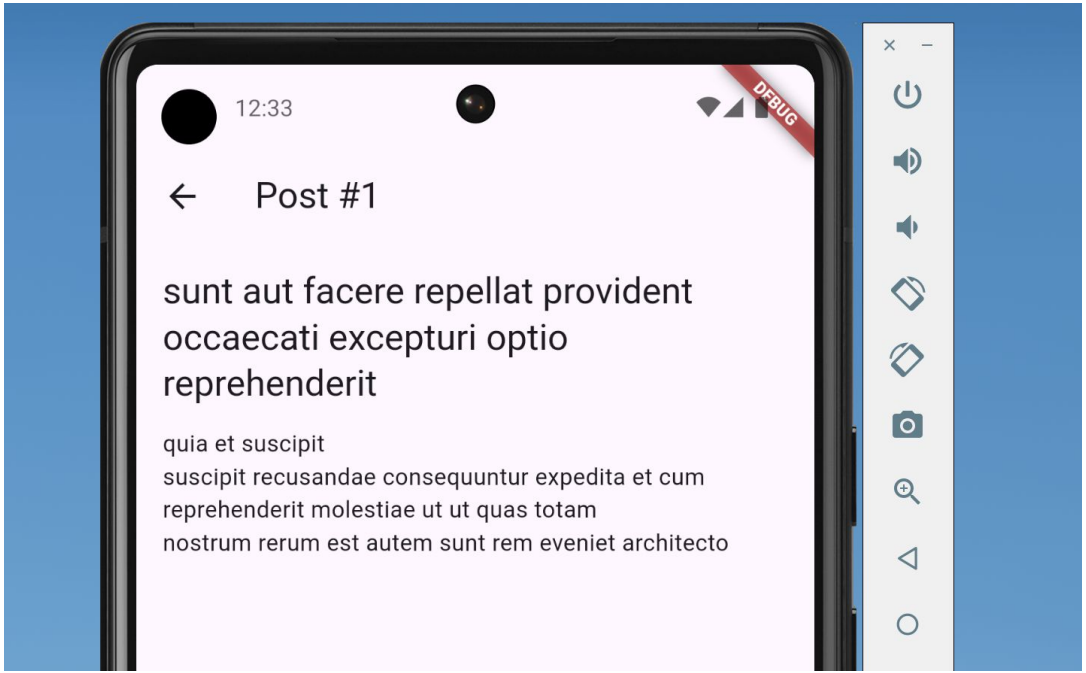
```dart
import 'package:flutter/material.dart';
import 'pages/post_list_page.dart';

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Post Reader',
      theme: ThemeData(useMaterial3: true),
      home: const PostListPage(),
    ); // MaterialApp
  }
}
```

สร้างไฟล์ app.dart



แก้ไขในไฟล์ lib/main.dart


12:30
DEBUG
Posts
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
Post #1
qui est esse
Post #2
ea molestias quasi exercitationem repellat qui ipsa sit aut
Post #3
eum et est occaecati
Post #4
nesciunt quas odio
Post #5
dolorem eum magni eos aperiam quia
Post #6
magnam facilis autem
Post #7
dolorem dolore est ipsam
Post #8
nesciunt iure omnis dolorem tempora et accusantium
Post #9
optio molestias id quia eum
Post #10


12:33
DEBUG
Post #1
sunt aut facere repellat provident occaecati excepturi optio reprehenderit
quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto

ทดสอบได้เลย !

https://docs.flutter.dev/cookbook/networking