



PERTEMUAN 10

HELPER MODUL DAN BLOC



1. Membuat Helper Modul



a. Menambahkan dependencies - 1

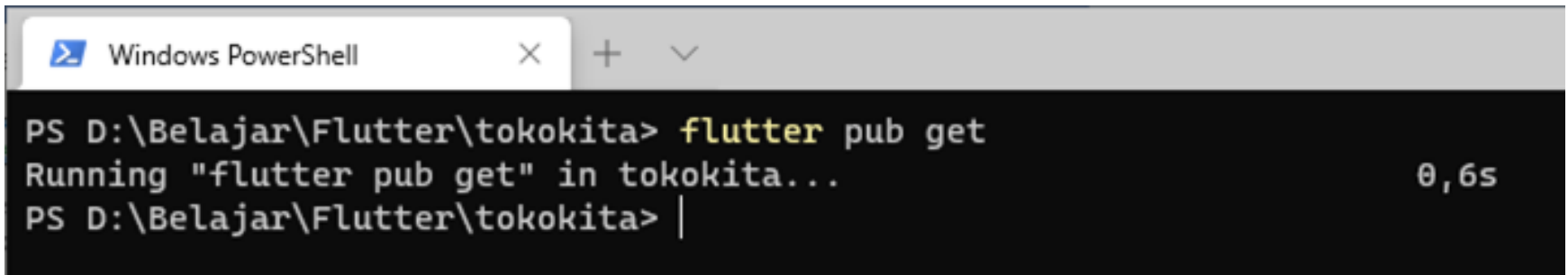
Dalam pembuatan aplikasi ini dibutuhkan depedensi untuk menerima dan mengirim request ke Rest API (**http**) dan depedensi untuk menyimpan token (**shared_preferences**). Pastikan komputer atau laptop terhubung ke internet, buka file **pubspec.yaml** kemudian tambahkan kode berikut

a. Menambahkan dependencies - 2

```
18. version: 1.0.0+1
19.
20. environment:
21.   sdk: ">=2.7.0 <3.0.0"
22.
23. dependencies:
24.   flutter:
25.     sdk: flutter
26.
27.
28.   # The following adds the Cupertino Icons font to your application.
29.   # Use with the CupertinoIcons class for iOS style icons.
30.   cupertino_icons: ^1.0.0
31.   http: ^0.12.2
32.   shared_preferences: ^0.5.12+2
33.
34. dev_dependencies:
35.   flutter_test:
36.     sdk: flutter
37.
```

a. Menambahkan dependencies - 3

Untuk mengunduh dependencies atau package yang telah ditambahkan, buka **CommandPrompt** kemudian masuk ke folder proyek dan ketikkan **flutter pub get** kemudian tekan Enter

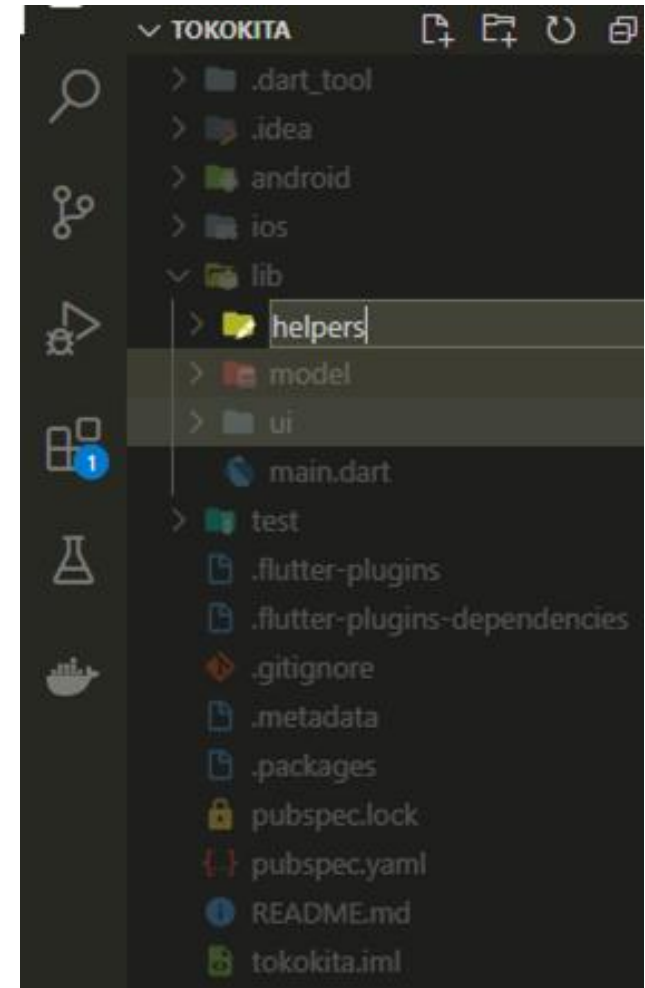


```
Windows PowerShell
PS D:\Belajar\Flutter\tokokita> flutter pub get
Running "flutter pub get" in tokokita... 0,6s
PS D:\Belajar\Flutter\tokokita> |
```

b. Membuat Class Token - 1

Buat sebuah folder dengan nama **helpers**

Kemudian pada folder **helpers** buat sebuah file dengan nama **user_info.dart** dan masukkan kode berikut



b. Membuat Class Token - 2

```
1. import 'package:shared_preferences/shared_preferences.dart';
2.
3. class UserInfo {
4.   Future setToken(String value) async {
5.     final SharedPreferences pref = await SharedPreferences.getInstance();
6.     return pref.setString("token", value);
7.   }
8.
9.   Future<String> getToken() async {
10.    final SharedPreferences pref = await SharedPreferences.getInstance();
11.    return pref.getString("token");
12.  }
13.
14.   Future setUserID(int value) async {
15.    final SharedPreferences pref = await SharedPreferences.getInstance();
16.    return pref.setInt("userID", value);
17.  }
18.
19.   Future<int> getUserID() async {
20.    final SharedPreferences pref = await SharedPreferences.getInstance();
21.    return pref.getInt("userID");
22.  }
23.
24.   Future logout() async {
25.    final SharedPreferences pref = await SharedPreferences.getInstance();
26.    pref.clear();
27.  }
28. }
```

c. Http request

(Membuat Modul Error Handling – 1)

Buat sebuah file pada folder **helpers** dengan nama **app_exception.dart** kemudian ketikkan kode berikut

```
1. class AppException implements Exception {  
2.   final _message;  
3.   final _prefix;  
4.  
5.   AppException([this._message, this._prefix]);  
6.  
7.   String toString() {  
8.     return "$_prefix$_message";  
9.   }  
10. }  
11.  
12. class FetchDataException extends AppException {  
13.   FetchDataException([String message])  
14.     : super(message, "Error During Communication: ");  
15. }  
16.
```


c. Http request (Membuat Modul Error Handling – 2)

```
16.  
17. class BadRequestException extends AppException {  
18.     BadRequestException([message]) : super(message, "Invalid Request: ");  
19. }  
20.  
21. class UnauthorisedException extends AppException {  
22.     UnauthorisedException([message]) : super(message, "Unauthorised: ");  
23. }  
24.  
25. class UnprocessableEntityException extends AppException {  
26.     UnprocessableEntityException([message])  
27.         : super(message, "Unprocessable Entity: ");  
28. }  
29.  
30. class InvalidInputException extends AppException {  
31.     InvalidInputException([String message]) : super(message, "Invalid Input: ");  
32. }
```

Pada file ini berfungsi sebagai penanganan jika terjadi error saat melakukan permintaan atau pengiriman ke Rest API

c. Http request

(Membuat Modul Http Request – 1)

Agar dapat mengirim atau menerima permintaan ke Rest API, dibuat sebuah fungsi baik itu method POST, GET dan DELETE.

Buat sebuah file di dalam folder **helpers** dengan nama **api.dart** dan masukkan kode berikut

```
1. import 'dart:io';  
2.  
3. import 'package:http/http.dart' as http;  
4. import 'package:tokokita/helpers/user_info.dart';  
5. import 'app_exception.dart';  
6.
```

c. Http request (Membuat Modul Http Request – 2)

```
6.
7. class Api{
8.     Future<dynamic> post(String url, dynamic data) async {
9.         var token = await UserInfo().getToken();
10.        var responseJson;
11.        try {
12.            final response = await http.post(url, body: data, headers: {
13.                HttpHeaders.authorizationHeader: "Bearer $token"
14.            });
15.            responseJson = _returnResponse(response);
16.        } on SocketException {
17.            throw FetchDataException('No Internet connection');
18.        }
19.        return responseJson;
20.    }
21.
22.    Future<dynamic> get(String url) async {
23.        var token = await UserInfo().getToken();
24.        var responseJson;
25.        try {
26.            final response = await http.get(url, headers: {
27.                HttpHeaders.authorizationHeader: "Bearer $token"
28.            });
29.            responseJson = _returnResponse(response);
30.        } on SocketException {
31.            throw FetchDataException('No Internet connection');
32.        }
33.        return responseJson;
34.    }
35.
```

c. Http request (Membuat Modul Http Request – 3)

```
35.
36. Future<dynamic> delete(String url) async {
37.     var token = await UserInfo().getToken();
38.     var responseJson;
39.     try {
40.         final response = await http.delete(url, headers: {
41.             HttpHeaders.authorizationHeader: "Bearer $token"
42.         });
43.         responseJson = _returnResponse(response);
44.     } on SocketException {
45.         throw FetchDataException('No Internet connection');
46.     }
47.     return responseJson;
48. }
49.
50. dynamic _returnResponse(http.Response response) {
51.     switch (response.statusCode) {
52.         case 200:
53.             return response;
54.         case 400:
55.             throw BadRequestException(response.body.toString());
56.         case 401:
57.         case 403:
58.             throw UnauthorisedException(response.body.toString());
59.         case 422:
60.             throw InvalidInputException(response.body.toString());
61.         case 500:
62.             default:
63.                 throw FetchDataException(
64.                     'Error occured while Communication with Server with StatusCode : ${response.statusCode}');
65.     }
66. }
67. }
```



c. Http request (Membuat List API url – 1)

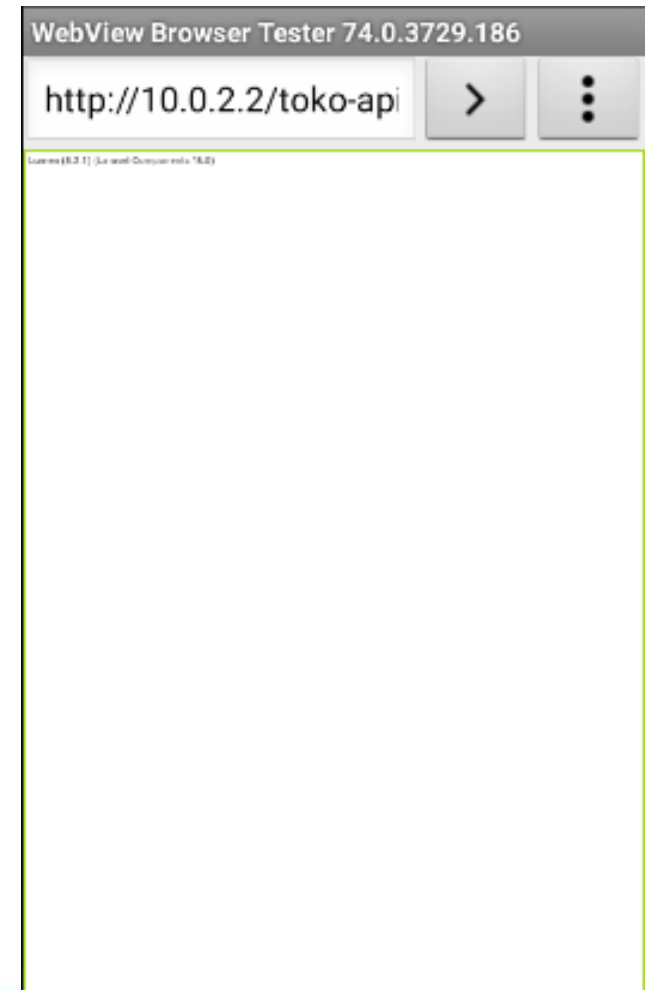
Buat sebuah file di dalam folder **helpers** dengan nama **api_url.dart**, dimana fungsi dari file ini adalah menyimpan alamat-alamat API yang telah dibuat sebelumnya (Endpoint dari API Spec)

c. Http request (Membuat List API url – 2)

```
1. class ApiUrl {
2.     static const String baseUrl = 'http://10.0.2.2/toko-api/public';
3.
4.     static const String registrasi = baseUrl + '/registrasi';
5.     static const String login = baseUrl + '/login';
6.     static const String listProduk = baseUrl + '/produk';
7.     static const String createProduk = baseUrl + '/produk';
8.
9.     static String updateProduk(int id) {
10.         return baseUrl + '/produk/' + id.toString() + '/update';
11.     }
12.
13.     static String showProduk(int id) {
14.         return baseUrl + '/produk/' + id.toString();
15.     }
16.
17.     static String deleteProduk(int id) {
18.         return baseUrl + '/produk/' + id.toString();
19.     }
20. }
```


c. Http request (Membuat List API url – 3)

Pada baris kedua (baseUrl) merupakan alamat IP dari Rest API, untuk memeriksa apakah terhubung dengan emulator atau tidak, dapat dilakukan dengan memasukkan alamat tersebut pada browser yang ada pada emulator atau handphone android yang terkoneksi dengan laptop

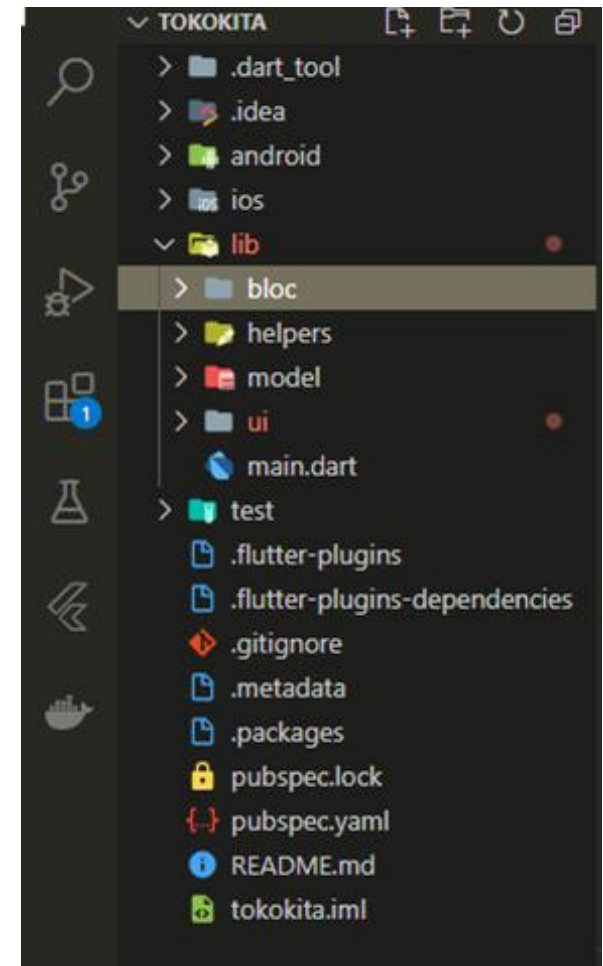




2. Membuat Bloc

2. Membuat Bloc

Buat sebuah folder bernama **bloc**. Di dalam folder ini berisi file-file yang berfungsi sebagai controller baik itu untuk melakukan proses login, registrasi dan lain-lain



a. Registrasi

Buat sebuah file dengan nama **registrasi_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'dart:convert';
2.
3. import 'package:tokokita/helpers/api.dart';
4. import 'package:tokokita/helpers/api_url.dart';
5. import 'package:tokokita/model/registrasi.dart';
6.
7. class RegistrasiBloc{
8.   static Future<Registrasi> registrasi({String nama, String email, String password})
9.     async {
10.       String apiUrl = ApiUrl.registrasi;
11.
12.       var body = {
13.         "nama": nama,
14.         "email":email,
15.         "password":password
16.       };
17.
18.       var response = await Api().post(apiUrl, body);
19.       var jsonObj = json.decode(response.body);
20.       return Registrasi.fromJson(jsonObj);
21. }
```

b. Login

Buat sebuah file dengan nama **login_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'dart:convert';
2.
3. import 'package:tokokita/helpers/api.dart';
4. import 'package:tokokita/helpers/api_url.dart';
5. import 'package:tokokita/model/login.dart';
6.
7. class LoginBloc{
8.   static Future<Login> login({String email, String password}) async {
9.     String apiUrl = ApiUrl.login;
10.    var body = {"email": email, "password": password};
11.    var response = await Api().post(apiUrl, body);
12.    var jsonObj = json.decode(response.body);
13.    return Login.fromJson(jsonObj);
14.  }
15. }
```

c. Logout

Buat sebuah file dengan nama **logout_bloc.dart** pada folder **bloc**. Kemudian masukkan kode berikut

```
1. import 'package:tokokita/helpers/user_info.dart';  
2.  
3. class LogoutBloc{  
4.   static Future logout() async {  
5.     await UserInfo().logout();  
6.   }  
7. }
```


d. Produk - 1

Pada bagian ini akan dibuat beberapa fungsi untuk mengambil, mengubah dan menghapus data produk dari Rest API. Buat sebuah file dengan nama **produk_bloc.dart** pada folder **bloc** kemudian masukkan kode berikut

```
1. import 'dart:convert';  
2.  
3. import 'package:tokokita/helpers/api.dart';  
4. import 'package:tokokita/helpers/api_url.dart';  
5. import 'package:tokokita/model/produk.dart';  
6.
```

d. Produk - 2

```
6.
7. class ProdukBloc{
8.   static Future<List<Produk>> getProduk() async {
9.     String apiUrl = ApiUrl.listProduk;
10.    var response = await Api().get(apiUrl);
11.    var jsonObj = json.decode(response.body);
12.    List<dynamic> listProduk = (jsonObj as Map<String, dynamic>)['data'];
13.    List<Produk> produk = [];
14.    for(int i=0; i < listProduk.length; i++){
15.      produk.add(Produk.fromJson(listProduk[i]));
16.    }
17.    return produk;
18.  }
19.
20.  static Future addProduk({Produk produk}) async {
21.    String apiUrl = ApiUrl.createProduk;
22.
23.    var body = {
24.      "kode_produk": produk.kodeProduk,
25.      "nama_produk": produk.namaProduk,
26.      "harga": produk.hargaProduk.toString()
27.    };
28.
29.    var response = await Api().post(apiUrl, body);
30.    var jsonObj = json.decode(response.body);
31.    return jsonObj['status'];
32.  }
33.}
```

d. Produk - 3

```
33.  
34. static Future<bool> updateProduk({Produk produk}) async {  
35.   String apiUrl = ApiUrl.updateProduk(produk.id);  
36.  
37.   var body = {  
38.     "kode_produk": produk.kodeProduk,  
39.     "nama_produk": produk.namaProduk,  
40.     "harga": produk.hargaProduk.toString()  
41.   };  
42.   print("Body : $body");  
43.   var response = await Api().post(apiUrl, body);  
44.   var jsonObj = json.decode(response.body);  
45.   return jsonObj['data'];  
46. }  
47.  
48. static Future<bool> deleteProduk({int id}) async {  
49.   String apiUrl = ApiUrl.deleteProduk(id);  
50.  
51.   var response = await Api().delete(apiUrl);  
52.   var jsonObj = json.decode(response.body);  
53.   return (jsonObj as Map<String, dynamic>)['data'];  
54. }  
55. }  
56.
```