MODUL 2

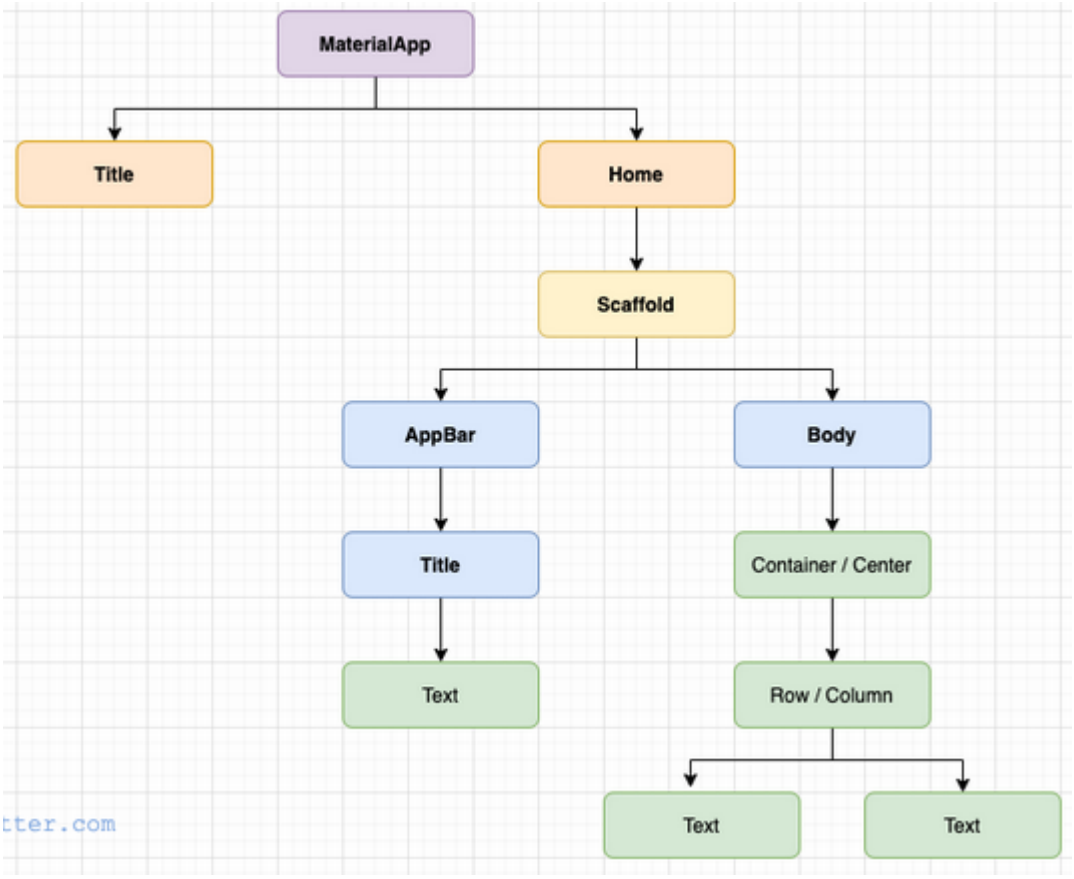
Mengenal Struktur Widget

Capaian Pembelajaran (CP) :

Memahami konsep serta menerapkan Integrated Development Environment (IDE)

Struktur Widget pada Aplikasi Flutter

Berikut adalah struktur widget minimal untuk aplikasi flutter.



Dapat kita lihat pada gambar diatas bahwa untuk Struktur Widget pada Aplikasi Flutter, umumnya membutuhkan widget MaterialApp.

Di dalam MaterialApp widget umumnya memiliki home properti yang dapat diisi oleh widget Scaffold dan widget lainnya sebagai child. Yang paling penting untuk diingat tentang flutter adalah semuanya adalah widget dan setiap widget memiliki kegunaannya masing-masing.

MaterialApp( )

MaterialApp adalah widget paling utama yang akan diakses pertama kali oleh fungsi main(). MaterialApp memiliki properti home yang dimana akan menjadi default route aplikasi.

```
1  void main() => MaterialApp(
2      home: Scaffold(),
3  );
```

Scaffold()

Scaffold adalah widget utama yang ditampilkan di aplikasi Anda. Ini pada dasarnya adalah wadah untuk semua widget lainnya. Jika Anda hanya memiliki satu widget di aplikasi pada awalnya, itu adalah Scaffold. Scaffold juga menampung beberapa fitur dasar aplikasi seperti AppBar, Body, BottomNavigationBar, FloatingActionButton, dll.

Untuk widget-widget dibawah dari scaffold, kita juga dapat mengklasifikasikannya menjadi tiga (3) kelompok, yaitu :

1. Standalone widget
2. Single child widget
3. Multiple children widget

Standalone widget

Standalone widget adalah jenis widget yang tidak mengandung widget lain. Mereka memiliki fungsi tertentu dan biasanya digunakan oleh jenis widget lainnya untuk mengisi konten, atau untuk sekedar menjadi styling properties. Contoh dari Standalone widget adalah :

- ✓ AppBar
- ✓ ImageAsset
- ✓ Icon
- ✓ Text
- ✓ TextStyle

```
1  Text("Belajar Flutter");
```

Single child widget

Sesuai namanya, singe child widget adalah widget yang hanya dapat memiliki SATU widget di dalamnya. Tipe widget ini memiliki properti yang bernama “child” digunakan untuk memasukan widget lain kedalam single child widget. Contoh dari single child widget yaitu :

- ✓ Center

- ✓ Container
- ✓ Expanded
- ✓ CircleAvatar
- ✓ RaisedButton
- ✓ Dll

```
Container(
  child: Text("Belajar Flutter"),
);
```

Multiple children widget

Widget ini dapat memiliki lebih dari satu widget di dalamnya. Tentu dalam setiap aplikasi pasti kita membutuhkan seperti kolom atau baris baik untuj layout ataupun konten. Untuk membuat hal seperti itu maka widget multiple children ini yang kamu butuhkan. Ciri dari widget ini yaitu memiliki properties yang bernama “children“. Contoh untuk multiple children widget yaitu :

- ✓ Row
- ✓ Column
- ✓ GridView
- ✓ Stack
- ✓ Dll

```
Column(
  children: <Widget>[
    Text('Ayo Belajar Flutter'),
    Text('di BelajarFlutter.com'),
  ],
)
```

Perbedaan Stateful dan Stateless Widget di Flutter

Sebelum mulai coba membuat aplikasi dengan Flutter, baiknya memahami dahulu konsep dasar dari komponen-komponennya. Dalam flutter semua komponen disebut dengan Widget. Widget memiliki dua tipe yaitu Stateful dan Stateless. Lalu apa perbedaan Stateful dan Stateless Widget pada aplikasi Flutter ?

Stateless Widget

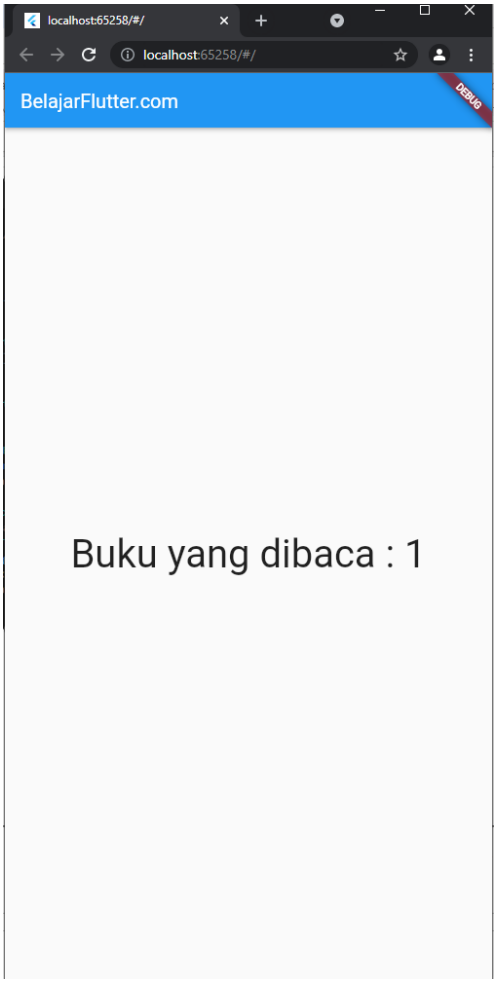
Secara sederhana Stateless Widget dapat diartikan sebagai Widget yang tidak dapat dirubah atau tidak akan pernah berubah.

Misal kita mempunya widget yang berisi text “Buku yang dibaca : 1” maka dari mulai text itu dibuat sampai aplikasi berjalan pun text tersebut tetap akan selalu menjadi “Buku yang dibaca : 1“.

main.dart widget_test.dart

lib > main.dart > ...

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

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
              title: const Text('BelajarFlutter.com'),
            ), // AppBar
            body: const Center(
              child: Text(
                'Buku yang dibaca : 1',
                style: TextStyle(fontSize: 40),
                textAlign: TextAlign.center,
              ), // Text
            ))); // Center // Scaffold // MaterialApp
  }
}
```



Contoh rill dalam penggunaan Stateless Widget biasanya yaitu untuk halaman “Tentang Aplikasi” yang berisi informasi nama atau logo aplikasi, versi dan lainya yang bersifat statis atau tidak perlu ada perubahan.

Stateful Widget

Stateful Widget merupakan widget yang dinamis atau dapat berubah. Berbanding terbalik dengan stateless, stateful widget dapat mengupdate tampilan, merubah warna, menambah jumlah baris dll. Jadi dapat disimpulkan bahwa **apapun widget yang dapat berubah** maka itulah stateful widget.

Merujuk pada contoh kasus sebelumnya dengan text “Buku yang dibaca : 1”, maka di stateful widget kita dapat merubah text tersebut sesuai kebutuhan. Berikut contoh code dan tampilan jika kita ingin membuat angka nya menjadi dinamis.

Latihan 1 (*untuk dikerjakan*) :

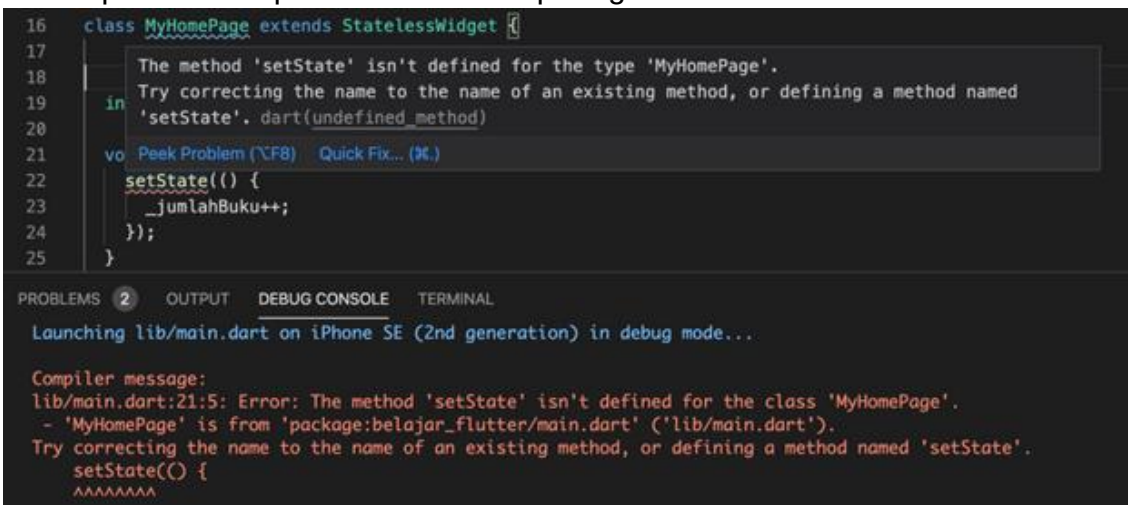
```
1  import 'package:flutter/material.dart';
2
3  void main() {
4  runApp(MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8  @override
9  Widget build(BuildContext context) {
10 return MaterialApp(
11 home: MyHomePage(),
12 );
13 }
14 }
15
16 class MyHomePage extends StatefulWidget {
17 @override
18  MyHomePageState createState() =>  MyHomePageState();
19 }
20
21 class _MyHomePageState extends State<MyHomePage> {
22 int _jumlahBuku = 1;
23
24 void _incrementCounter() {
25 setState(() {
26 _jumlahBuku++;
27 });
28 }
29
30 @override
31 Widget build(BuildContext context) {
32
33 return Scaffold(
34 appBar: AppBar(
35 title: Text('BelajarFlutter.com'),
36 ),
37 body: Center(
38 child: Column(
39
```

```
56  mainAxisAlignment: MainAxisAlignment.center,
57
58  children: <Widget>[
59
60  Text(
61
62  'Buku yang dibaca : $_jumlahBuku',
63
64  style: TextStyle( fontSize: 40 ),
65
66  textAlign: TextAlign.center,
67
68  ),
69
70  ],
71
72  ),
73
74  ),
75
76  floatingActionButton: FloatingActionButton(
77
78  onPressed: _incrementCounter,
79
80  tooltip: 'Tambah',
81
82  child: Icon(Icons.add),
83
84  ),
    );
    }
}
```

Dapat kita lihat di code baris ke 25 yaitu penggunaan fungsi **setState()** yang akan bertugas untuk memberitahu widget bahwa ada object yang berubah pada State, sehingga akan melakukan build ulang pada Widget tersebut.

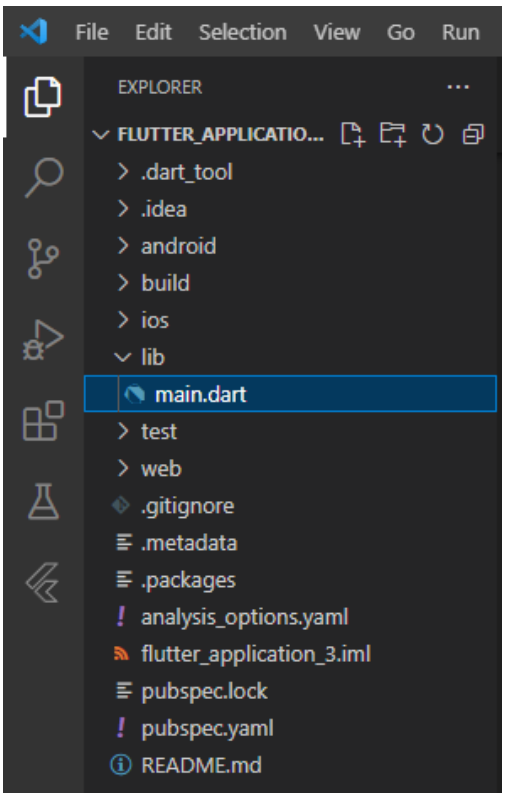
Apakah bisa menggunakan fungsi setState() di Stateless Widget?

Tidak, karena apabila memanggil fungsi setState() di stateless widget maka akan muncul pesan error pada text editor seperti gambar di bawah ini



Struktur Direktori Project Flutter

Ini adalah struktur direktori project Flutter :



Penjelasan:

- `android` berisi source code untuk aplikasi android;
- `ios` berisi source code untuk aplikasi iOS;
- `lib` berisi source code Dart, di sini kita akan menulis kode aplikasi;
- `test` berisi source code Dart untuk testing aplikasi;
- `.gitignore` adalah file Git;
- `.metadata` merupakan file yang berisi metadata project yang di-generate otomatis;
- `.packages` merupakan file yang berisi alamat path package yang dibuat oleh pub;
- `flutter_app.iml` merupakan file XML yang berisi keterangan project;
- `pubspec.lock` merupakan file yang berisi versi-versi library atau package. File ini dibuat oleh pub. Fungsinya untuk mengunci versi package.
- `pubspec.yaml` merupakan file yang berisi informasi tentang project dan libraray yang dibutuhkan;
- `README.md` merupakan file markdown yang berisi penjelasan tentang source code.

Struktur Dasar Kode Aplikasi

Saat kamu membuat project baru, kode program awal yang akan kita dapatkan pada main.dart akan seperti ini :

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title}) : super(key: key);

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
```

```dart
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
          // arranges them vertically. By default, it sizes itself to fit its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Invoke "debug painting" (press "p" in the console, choose the
          // "Toggle Debug Paint" action from the Flutter Inspector in Android
          // Studio, or the "Toggle Debug Paint" command in Visual Studio Code)
          // to see the wireframe for each widget.
          //
          // Column has various properties to control how it sizes itself and
          // how it positions its children. Here we use mainAxisAlignment to
          // center the children vertically; the main axis here is the vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

Panjang sekali…

Sebenarnya kode ini sangat sederhana, yang membuat ia panjang adalah komentar-komentarnya.

Struktur kode di atas sebenarnya terdiri dari tiga bagian:

1. Bagian import;
2. Bagian main;
3. Bagian widget.

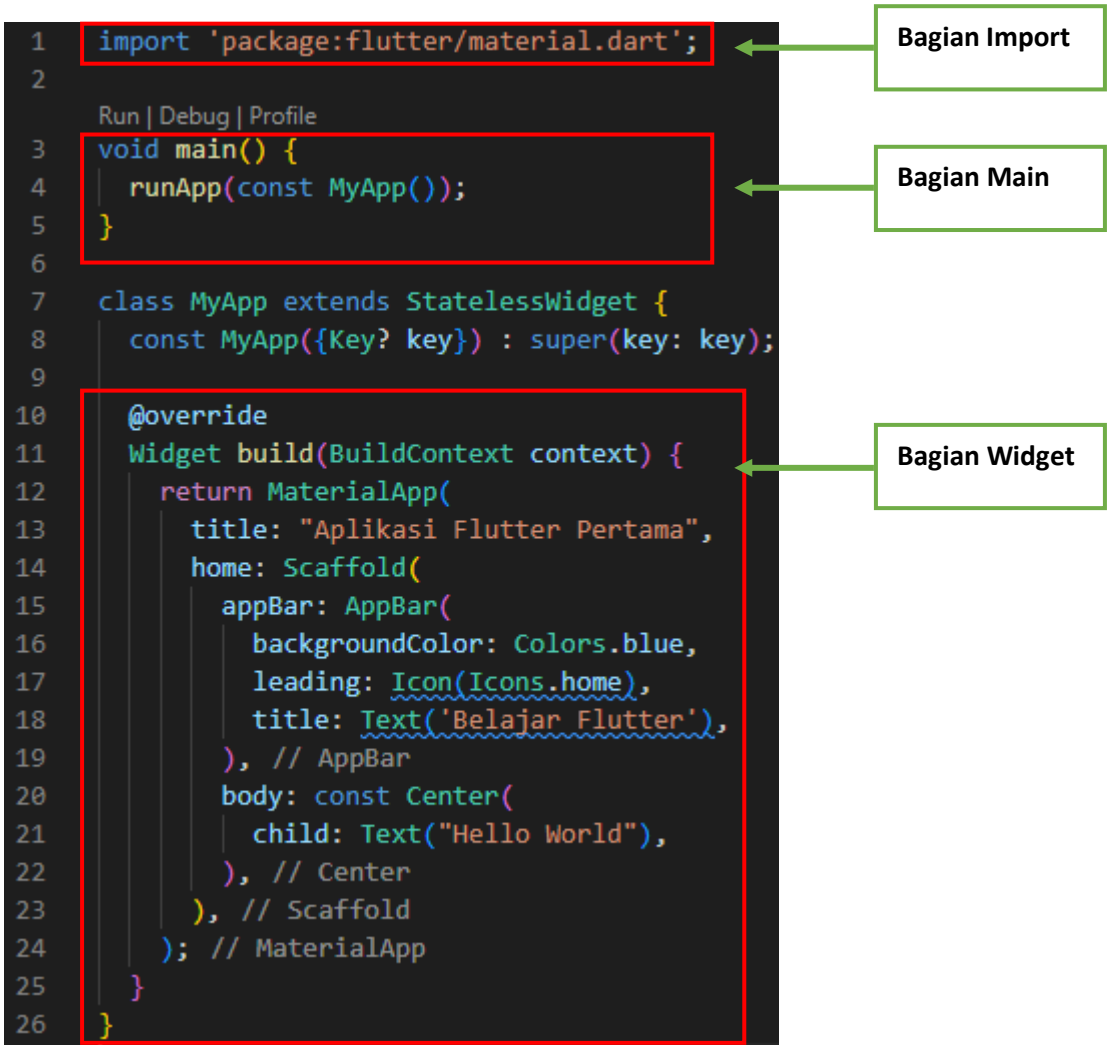
Mari kita lihat contoh kode yang sama.

Latihan 2 (*untuk dikerjakan*) :

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Aplikasi Flutter Pertama",
      home: Scaffold(
```

```
        appBar: AppBar(
          title: Text('Belajar Flutter'),
        ),
        body: const Center(
          child: Text("Hello World"),
        ),
      ),
    );
  }
}
```

Coba perhatikan!




```
1  import 'package:flutter/material.dart';
2
   Run | Debug | Profile
3  void main() {
4    runApp(const MyApp());
5  }
6
7  class MyApp extends StatelessWidget {
8    const MyApp({Key? key}) : super(key: key);
9
10   @override
11   Widget build(BuildContext context) {
12     return MaterialApp(
13       title: "Aplikasi Flutter Pertama",
14       home: Scaffold(
15         appBar: AppBar(
16           backgroundColor: Colors.blue,
17           leading: Icon(Icons.home),
18           title: Text('Belajar Flutter'),
19         ), // AppBar
20         body: const Center(
21           child: Text("Hello World"),
22         ), // Center
23       ), // Scaffold
24     ); // MaterialApp
25   }
26 }
```

1. Bagian Import

Bagian import adalah tempat kita mendeklarasikan atau mengimpor library yang dibutuhkan pada aplikasi.

2. Bagian Main

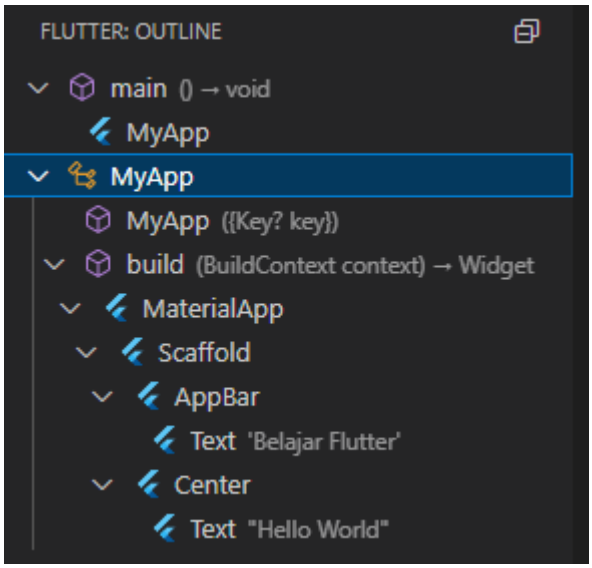
Bagian main adalah fungsi utama dari aplikasi yang akan menjadi *entri point*. Fungsi ini akan dieksekusi pertama kali saat aplikasi dibuka.

3. Bagian Widget

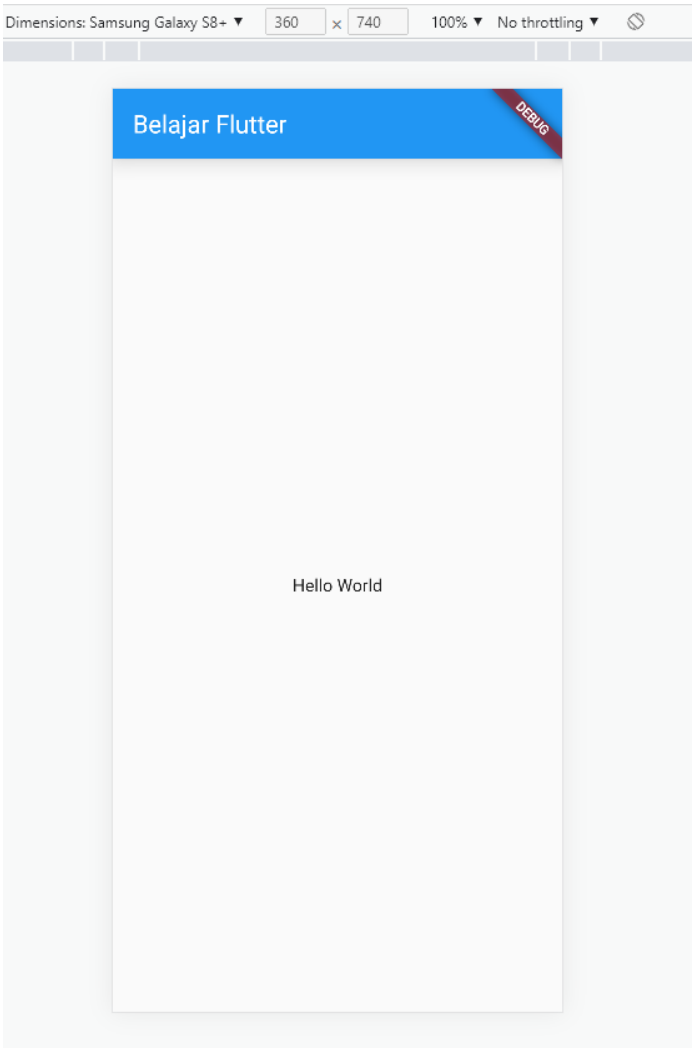
Bagian widget adalah tempat kita membuat widget. Aplikasi Flutter sebenarnya terdiri dari susunan widget. Widget bisa kita bilang elemen-elemen seperti Tombol, Teks, Layaout, Image, dan sebagainya.

Sebuah widget dapat berisi widget.

Pada kode program di atas, kita menggunakan beberapa widget yang tersusun seperti ini:



Pada aplikasi akan ditampilkan seperti ini:



Kesimpulan

Jika kita ingin membuat komponen sederhana dan bersifat statis maka Stateless Widget adalah pilihan yang tepat. Tetapi apabila membutuhkan komponen yang dinamis maka gunakan stateful widget.

Sumber :

https://belajarflutter.com/perbedaan-stateful-dan-stateless-widget-di-flutter/

https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html

https://www.petanikode.com/flutter-dasar/

Tugas Mandiri :

Kerjakan **Latihan 1** dan **Latihan 2** pada modul ini, dan kumpulkan sesuai dengan instruksi dari guru pengajar.