



# Modul Praktikum **Pemrograman Mobile**



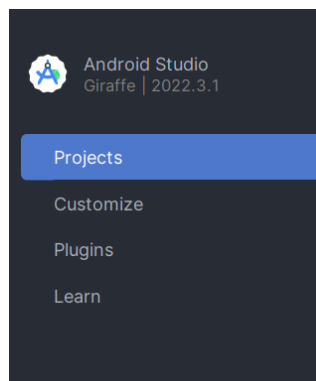
# MEMULAI PROJECT & WIDGET DASAR - PART I

## MEMULAI PROJECT FLUTTER

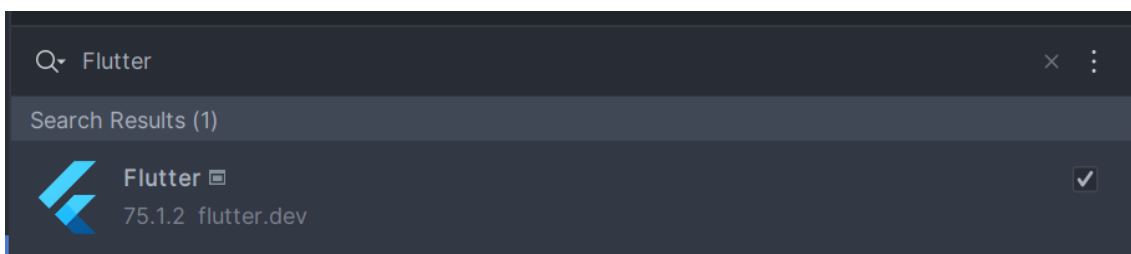
Perlu diketahui bahwa Flutter adalah sebuah framework pengembangan cross-platform application yang dikembangkan oleh Google. Namun, pertanyaan utama adalah *"bagaimana kita memulai proyek Flutter?"*. Pada kesempatan kali ini, kita akan membuat project Flutter pertama kita dengan menggunakan 2 code editor yang biasa digunakan oleh para *sepuh* pengembang aplikasi!

## ANDROID STUDIO

Untuk memulai project Flutter menggunakan Android Studio, kita perlu menginstal plugin Flutter terlebih dahulu. Pergi ke halaman *Welcome*, lalu klik 'Plugins' pada sidebar.

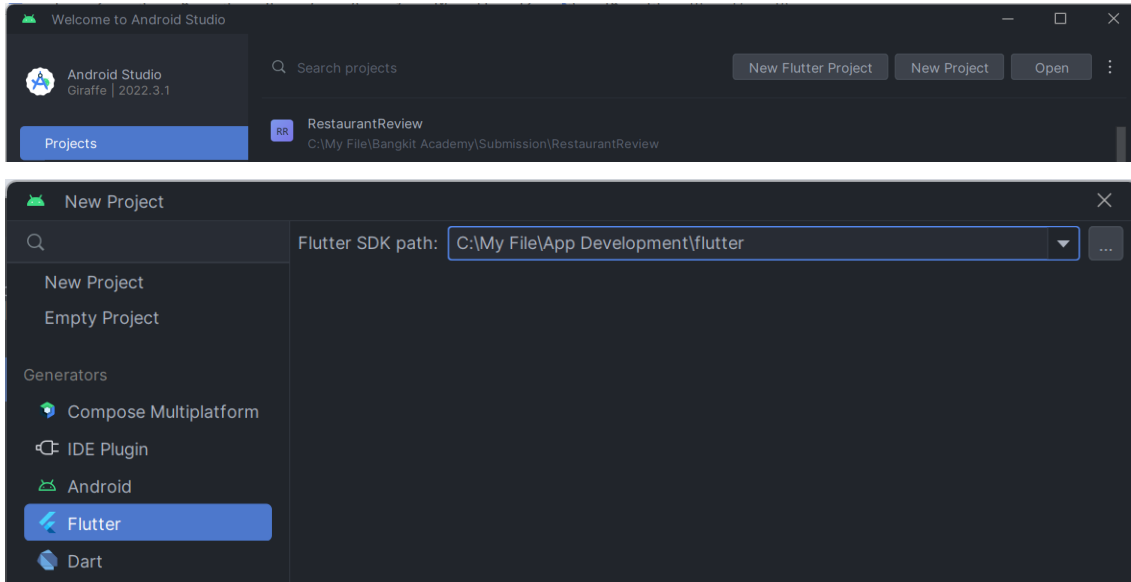


Lalu ketik "Flutter" pada kolom input pencarian, lalu unduh plugin Flutter.. Plugin Tersebut juga akan meminta kita untuk mengunduh plugin Dart sebagai plugin tambahannya.

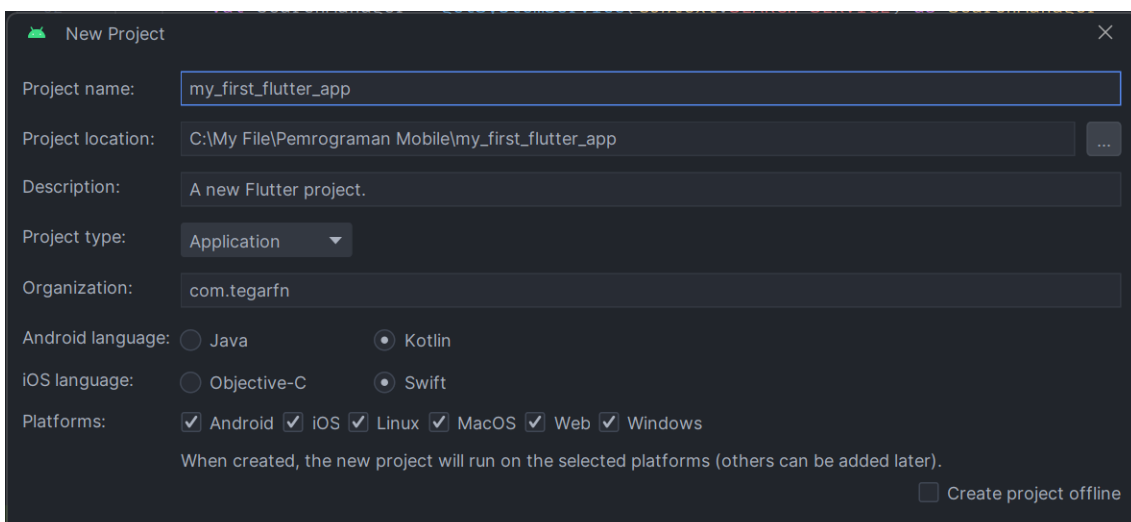




Setelah selesai mengunduh, kembali ke halaman *project*, dan akan ada tombol baru bertuliskan “*New Flutter Project*”. Klik tombol tersebut, dan Android Studio akan meminta *path* dimana kita meletakkan Flutter SDK yang telah kita simpan sebelumnya.



Lalu masukkan nama project, nama project harus menggunakan huruf kecil. Pilih direktori yang digunakan, kita perlu membuat sebuah direktori khusus sebagai direktori aplikasi Flutter. Tentukan tipe project yang akan dibuat, kita cukup memilih “*Application*”. Dan terakhir tentukan platform apa yang akan kita gunakan. Selain dari keempat itu, bisa kita abaikan.





Tunggu sejenak hingga proses selesai, dan project Flutter pertama kita berhasil terbuat!

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

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


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

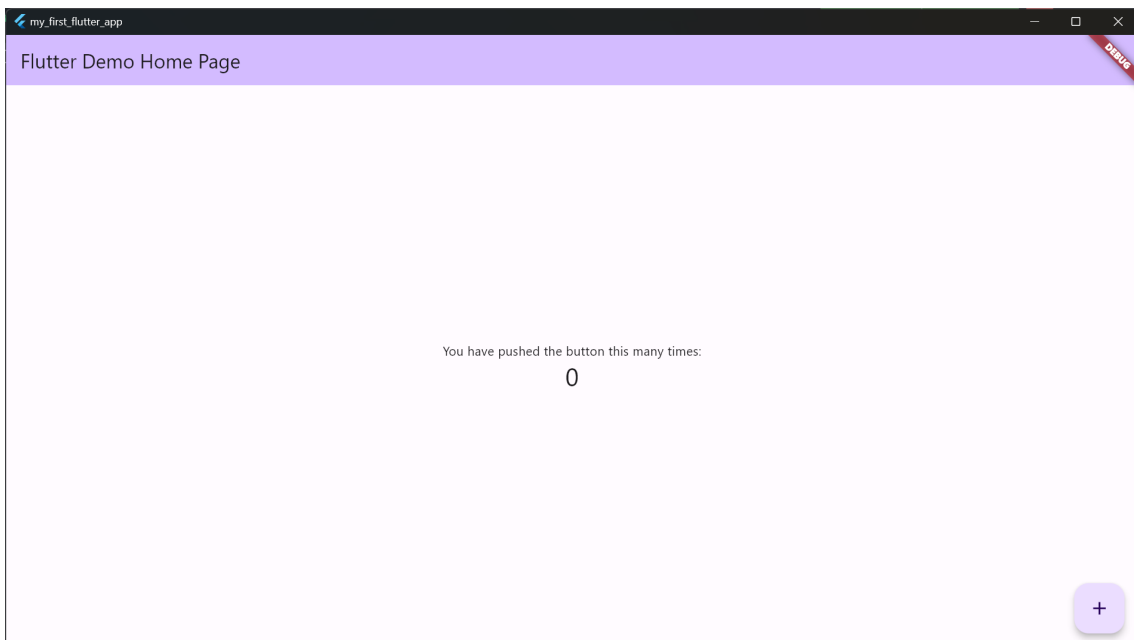
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a blue toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        // Notice that the counter didn't reset back to zero; the application
        // state is not lost during the reload. To reset the state, use hot
        // restart instead.
        // This works for code too, not just values: Most code changes can be
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}

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

  // 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.

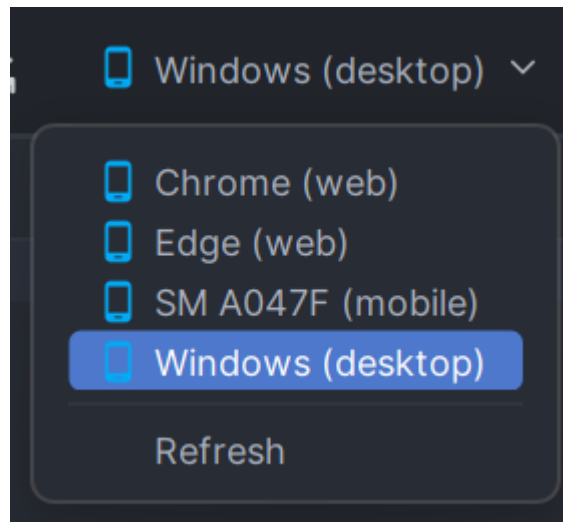
  // This class is the configuration for the state. It holds the value for this
```

Untuk memulai debugging dan menjalankan aplikasi kita, anda bisa menggunakan shortcut `shift+F10` atau tekan tombol . Pada contoh kali ini, aplikasi Flutter berjalan sebagai *Desktop App*.



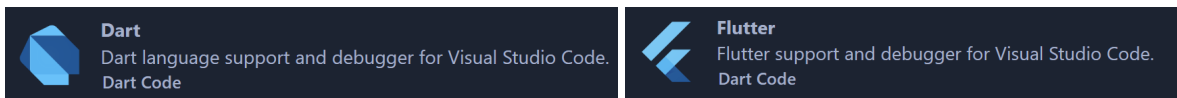


Kita dapat mengganti platform dengan menekan tombol device di bagian atas kanan.

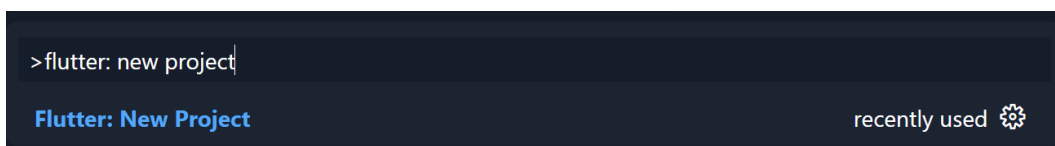


## VISUAL STUDIO CODE

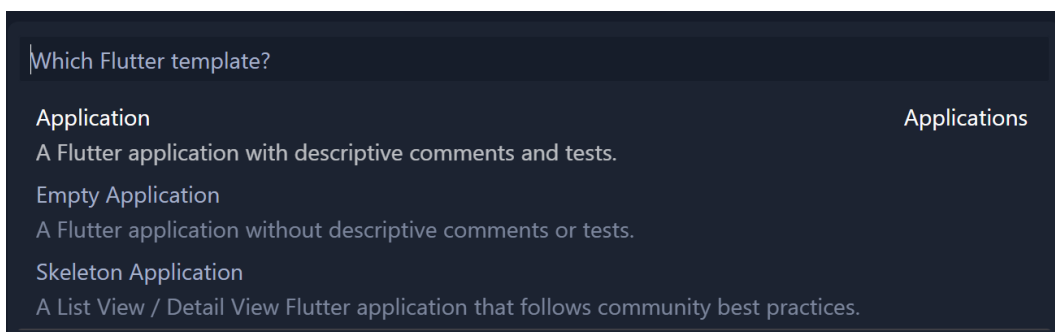
Jika kita menggunakan Visual Studio Code sebagai Code Editor, maka anda perlu menginstal beberapa ekstensi yaitu Dart dan Flutter di Visual Studio Code kita.



Setelah menginstal kedua ekstensi tersebut, tekan `ctrl+shift+P`, lalu ketik *"flutter: new project"* dan tekan Enter.

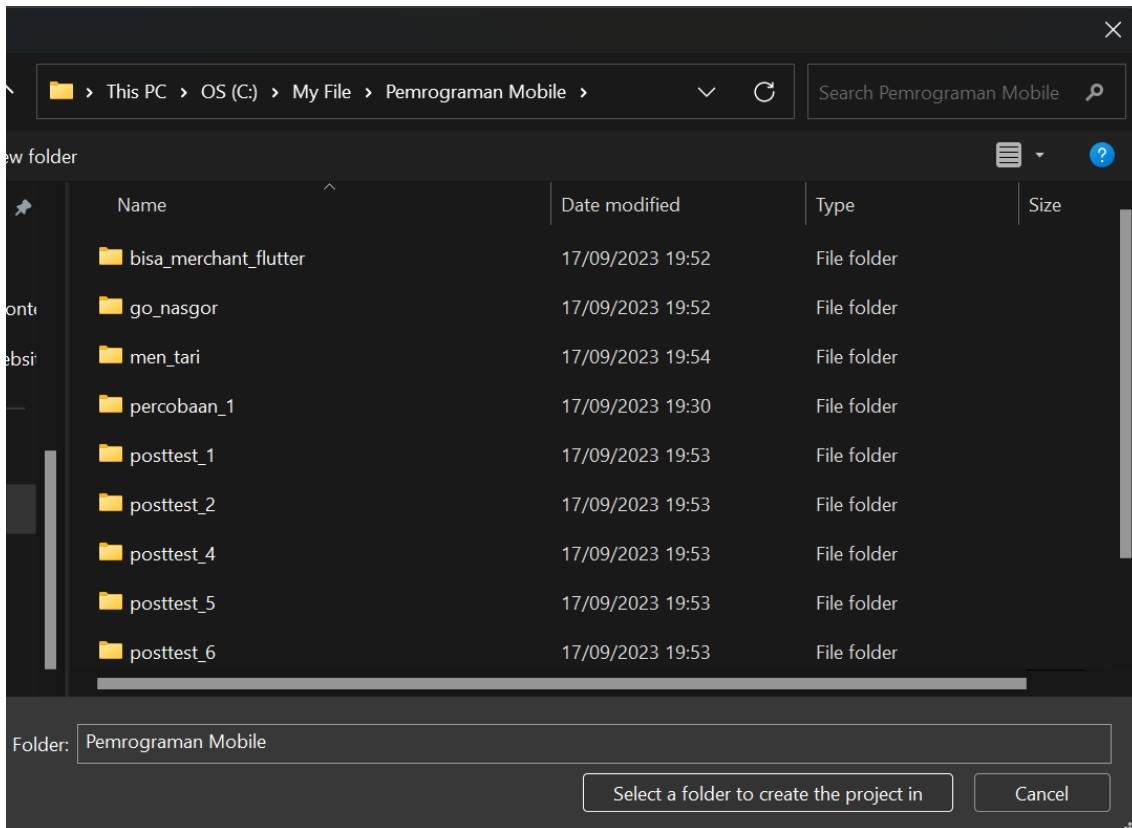


Lalu pilih tipe project yang akan kita buat. Kita cukup memilih *"Application"*.

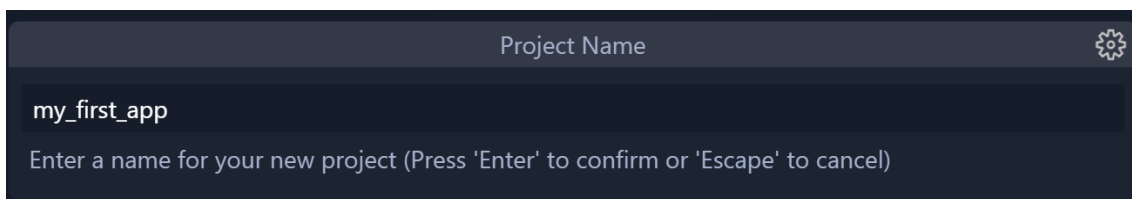




Selanjutnya, pilih direktori sebagai tempat project Kita. Pada contoh kali ini, kita membuat sebuah folder baru yang akan menyimpan seluruh project Flutter kita agar tersusun rapi dan mudah untuk dicari.



Lalu tentukan nama project kita, yang sekaligus menjadi folder dimana project aplikasi kita tersimpan. Project Flutter mengharuskan penamaan project dengan huruf kecil dan '\_' (underscore) jika ingin membuat spasi.

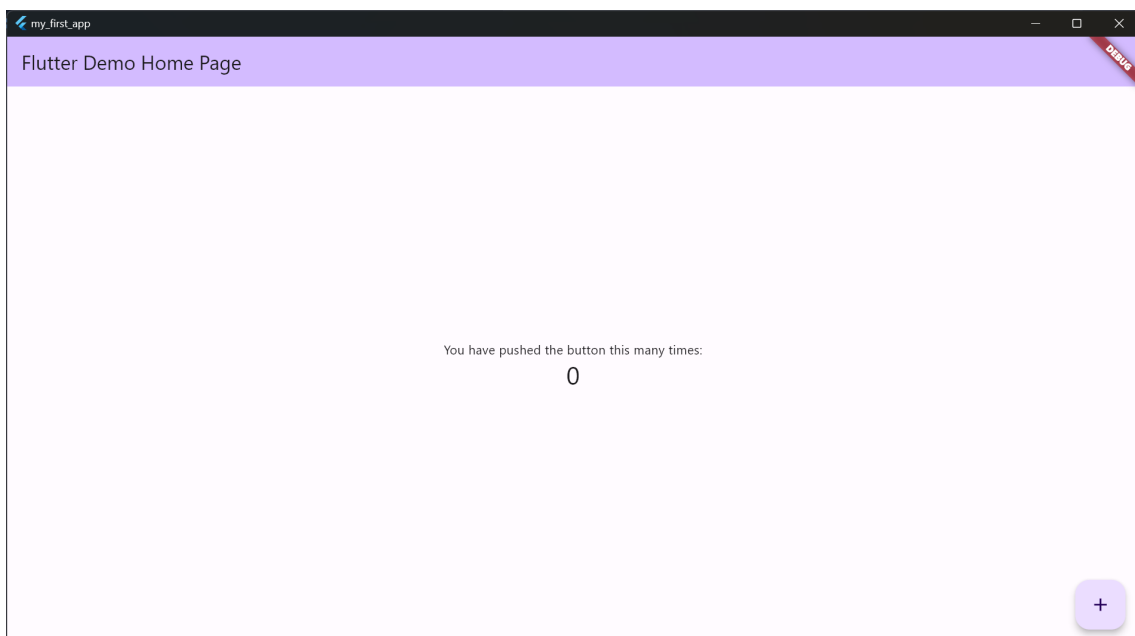




Tunggu hingga proses selesai, dan *voilà!* Project Flutter pertama kita terbentuk di Visual Studio Code!

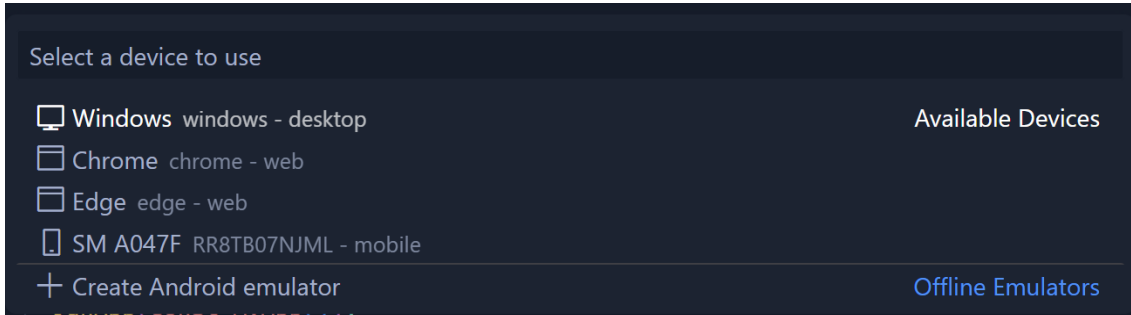
```
lib > main.dart > MyApp > build
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  // This widget is the root of your application.
11  @override
12  Widget build(BuildContext context) {
13    return MaterialApp(
14      title: 'Flutter Demo',
15      theme: ThemeData(
16        // This is the theme of your application.
17        // TRY THIS: Try running your application with "flutter run". You'll see
18        // the application has a blue toolbar. Then, without quitting the app,
19        // try changing the seedColor in the colorScheme below to Colors.green
20        // and then invoke "hot reload" (save your changes or press the "hot
21        // reload" button in a Flutter-supported IDE, or press "r" if you used
22        // the command line to start the app).
23        //
24        // Notice that the counter didn't reset back to zero; the application
25        // state is not lost during the reload. To reset the state, use hot
26        // restart instead.
27        //
28        // This works for code too, not just values: Most code changes can be
29        // tested with just a hot reload.
30        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
31        useMaterial3: true,
32      ),
33    );
34    home: const MyHomePage(title: 'Flutter Demo Home Page'),
35  };
36 }
```

Untuk menjalankan aplikasinya, cukup perlu menekan F5 maka Flutter akan melakukan debugging dan menampilkan aplikasi kita! Pada contoh kali ini, project Flutter kita berjalan sebagai *Desktop App*.





Kita juga dapat merubah perangkat apa yang akan digunakan dengan menekan tombol bertuliskan “Windows (...)” pada pojok kanan bawah. kita bisa menggunakan browser, perangkat eksternal, dan juga emulator yang sudah kita unduh di Android Studio.







## WIDGET DASAR #1

Apa itu Widget? Widget adalah semua komponen seperti button, text, icon dan sebagainya yang membentuk sebuah tampilan atau kerangka aplikasi. Widget adalah bagian-bagian kecil yang dapat digabungkan dan disusun untuk membentuk tampilan aplikasi kita. Widget dalam Flutter dapat berupa widget tunggal, seperti teks atau gambar, atau widget yang lebih kompleks seperti tata letak (layout) atau bahkan seluruh tampilan aplikasi.

Flutter menggunakan sebuah desain sistem yang disebut "Material Design" yang mayoritas digunakan oleh produk Android dan "Cupertino" yang digunakan oleh produk Apple untuk membangun tampilan aplikasi yang konsisten dan menarik di berbagai platform.

Kali ini kita akan mempelajari cara menggunakan Text, Container, dan Layouting menggunakan Row & Column.

## TEXT

Widget text adalah bagian yang sangat penting dari sebuah UI aplikasi mobile mana pun. Pada Flutter, untuk menampilkan text kita dapat menggunakan Text widget. Text widget mempunyai beberapa properti dasar seperti:

- **Style** → berfungsi untuk mengubah styling text
- **TextAlign** → berfungsi untuk mengubah posisi text
- **TextDirection** → berfungsi untuk mengubah arah text

Contoh kita akan membuat text dengan style ukuran font 20px dan kita ubah font weightnya menjadi Bold. Maka contoh codingannya akan seperti ini:

```
body: Center(  
  child: Text(  
    "Halo Para Praktikan!",  
    style: TextStyle(  
      fontSize: 20,  
      fontWeight: FontWeight.bold,  
      color: Colors.green,  
    ),  
  ),  
)
```



Hasil:

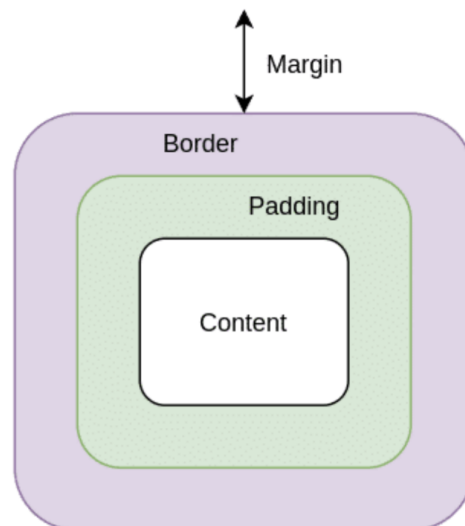
Halo Para Praktikan!

## CONTAINER

Widget container adalah sebuah widget yang bertugas sebagai pembungkus dari widget lain. Dinamakan container karena nantinya, ia bisa memiliki properti-properti bingkai seperti:

- padding
- margin
- border
- border radius
- box shadow
- dan lain-lain

Ilustrasi:



Pada ilustrasi di atas, kotak berwarna putih adalah child widget. Sedangkan semuanya dari margin, border, dan padding adalah widget container.



Contoh kita akan membuat container dengan lebar dan tinggi 200x200 piksel dan beri warna merah pada background container.

```
body: Center(  
  child: Container(  
    width: 200,  
    height: 200,  
    color: Colors.red,  
  ),  
)
```

Hasil:



Widget Container juga memiliki property decoration untuk memperbagus tampilan container. Property decoration menggunakan widget yang BoxDecoration. Contoh kita akan menambahkan border dan border radius pada container kita sebelumnya.

**Note:** Jika container menggunakan property decoration, maka property color/warna harus dipindah ke dalam BoxDecoration, jika tidak maka container akan error.

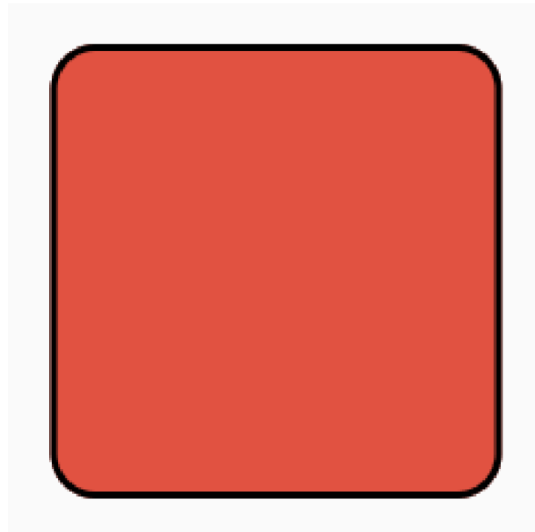
```
body: Center(  
  child: Container(  
    width: 200,  
    height: 200,  
    decoration: BoxDecoration(  
      color: Colors.red,  
      borderRadius: BorderRadius.circular(20),  
      border: Border.all(  
        color: Colors.black,  

```



```
        width: 3,  
      ),  
    ),  
  ),  
)
```

Hasil:



Tujuan utama membuat container adalah untuk membungkus konten atau child di dalam container tersebut. Jadi kita akan menambahkan sebuah konten berupa widget Text ke dalam container tersebut.

```
body: Center(  
  child: Container(  
    width: 200,  
    height: 200,  
    alignment: Alignment.center,  
    decoration: BoxDecoration(  
      color: Colors.red,  
      borderRadius: BorderRadius.circular(20),  
      border: Border.all(  
        color: Colors.black,  
        width: 3,  
      ),  
    ),  
  ),  
  child: Text("Halo Para Praktikan!"),  
)
```



Perhatikan, kita menambahkan property child dan alignment pada container tersebut. Property child berisi konten di dalam container tersebut dan property alignment berfungsi untuk meletakkan di mana child atau konten ingin ditaruh. Contoh kita ingin menaruh text di tengah-tengah container, maka kita atur alignmentnya menjadi center.



## ROW & COLUMN

**Row** adalah layout widget yang mengatur widget-widget yang di dalamnya menjadi horizontal.

Contoh:

```
body: Center(  
  child: Row(  
    children: [  
      Container(  
        width: 100,  
        height: 100,  
        margin: EdgeInsets.only(right: 10),  
        color: Colors.red,  
      ),  
      Container(  
        width: 100,  
        height: 100,  
        margin: EdgeInsets.only(right: 10),  
        color: Colors.blue,  
      ),  
      Container(  

```



```
        width: 100,  
        height: 100,  
        margin: EdgeInsets.only(right: 10),  
        color: Colors.green,  
      ),  
    ],  
  ),  
,
```

Hasil:



Berbeda dengan Row, **Column** adalah layout widget yang mengatur widget-widget yang di dalamnya menjadi Vertikal.

Contoh:

```
body: Center(  
  child: Column(  
    children: [  
      Container(  
        width: 100,  
        height: 100,  
        margin: EdgeInsets.only(bottom: 10),  
        color: Colors.red,  
      ),  
      Container(  
        width: 100,  
        height: 100,  
        margin: EdgeInsets.only(bottom: 10),  
        color: Colors.blue,  
      ),  
      Container(  
        width: 100,  
        height: 100,  
      ),  
    ],  
  ),  
)
```





## BERBAGI PROJECT FLUTTER

Sering kali asisten lab mengingatkan “*Jangan lupa di clean!*” ataupun kita dihadapi permasalahan “*kok upload file flutter ke GitHub lama banget?*”, apa yang sebenarnya terjadi? Itu dikarenakan Flutter menyimpan banyak sekali file dependensi, cache, dan file sementara lainnya. Untuk mengatasi hal tersebut, setiap kali kita mengakhiri untuk mengembangkan aplikasi kita, kita perlu ‘membersihkan’ direktori tempat aplikasi Flutter kita berada dengan menggunakan command `flutter clean` pada terminal. Dengan begitu, aplikasi kita akan lebih mudah untuk di compress/export dan dibagikan.

```
PS C:\My File\Pemrograman Mobile\my_first_app> flutter clean
Deleting build... 19ms
Deleting .dart_tool... 1ms
Deleting Generated.xcconfig... 0ms
Deleting flutter_export_environment.sh... 0ms
Deleting ephemeral... 0ms
Deleting ephemeral... 0ms
Deleting ephemeral... 4ms
```

Setelah kita memasukkan command tersebut, maka akan muncul banyak error karena beberapa file hilang dari direktori aplikasi Flutter kita. Untuk mengembalikannya, gunakan command `flutter pub get`, dengan begitu, Flutter akan menginstal semua file yang terdapat pada dependensi dan file meta lainnya.

```
PS C:\My File\Pemrograman Mobile\my_first_app> flutter pub get
Resolving dependencies...
  collection 1.17.2 (1.18.0 available)
  material_color_utilities 0.5.0 (0.8.0 available)
  meta 1.9.1 (1.10.0 available)
  stack_trace 1.11.0 (1.11.1 available)
  stream_channel 2.1.1 (2.1.2 available)
  test_api 0.6.0 (0.6.1 available)
Got dependencies!
```