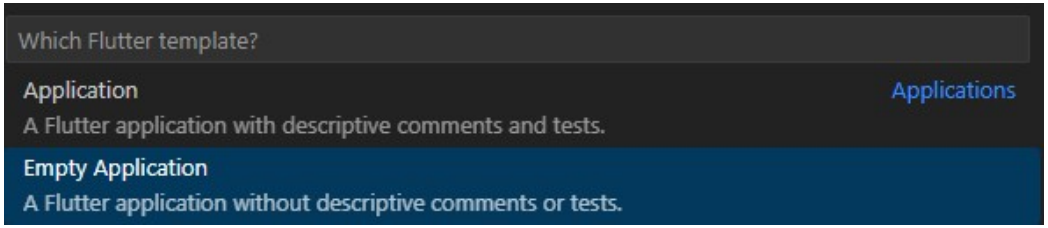
Project template “Application”

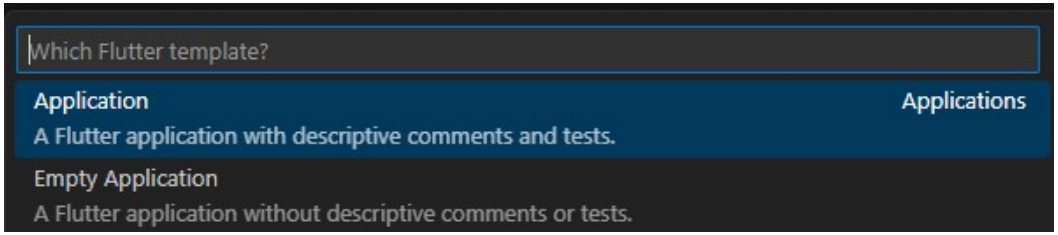
- Be able to use Flutter‘s project template „Application“
- Know how to delete all comment lines in a Dart source file

Project template „Application“

Till now we used the project template “Empty Application”:



Let’s now try “Application”:



Compare first lines in main.dart (without comments)

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

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World!'),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

App(lication) and HomePage are separated (to “open the door” for more pages)

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

Run | Debug | Profile
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(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}

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

  final String title;

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

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

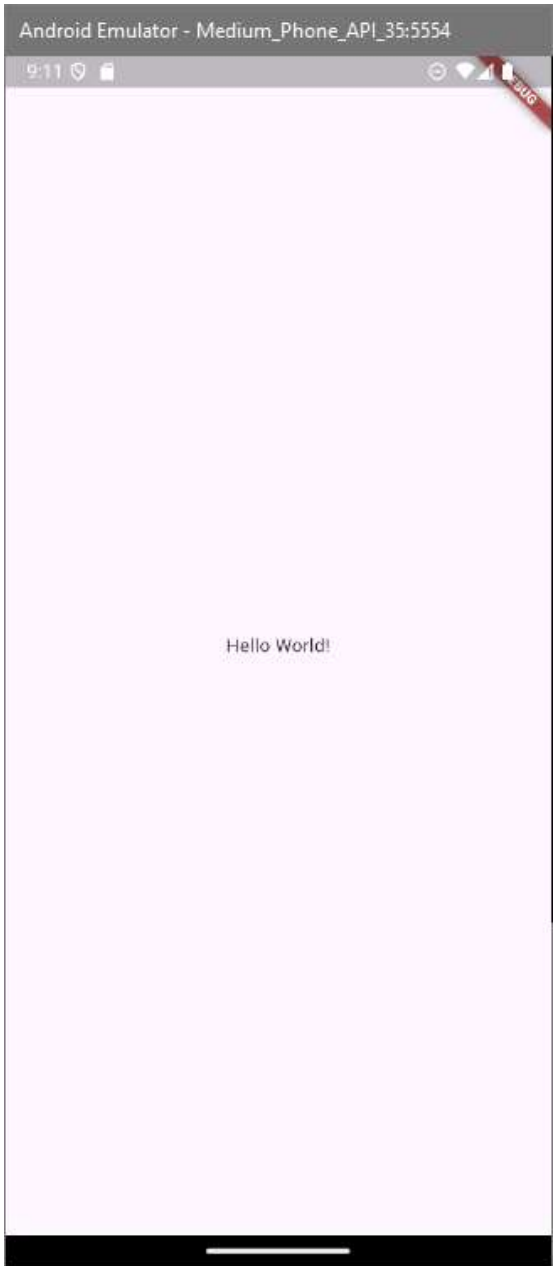
Compare the build methods (without comments)

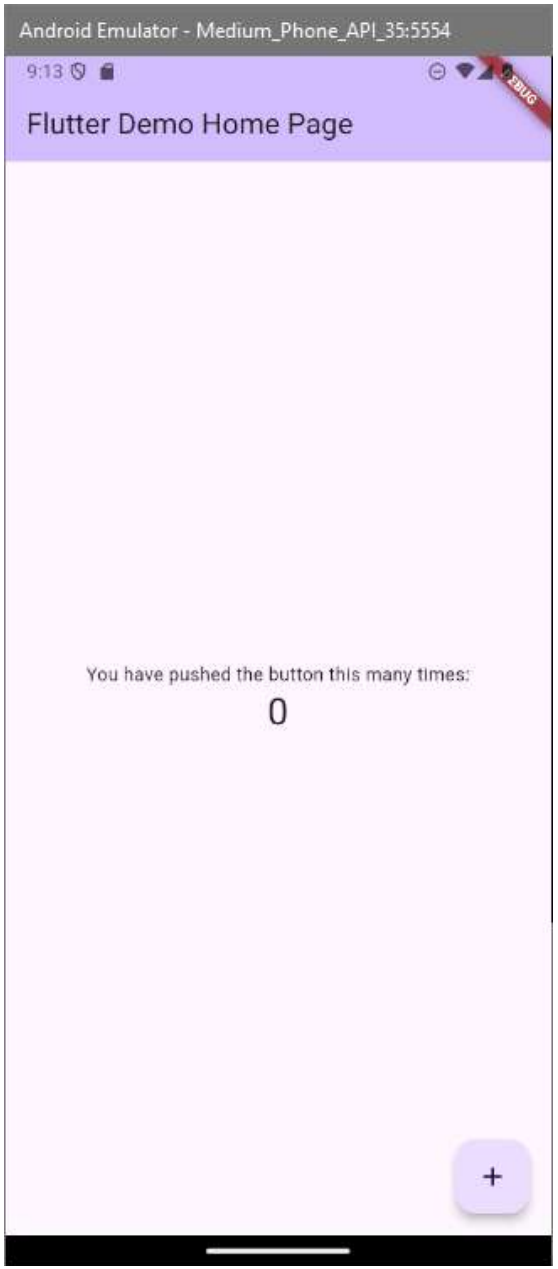
```dart
import 'package:flutter/material.dart';

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Scaffold(
        body: Center(
          child: Text('Hello World!'),
        ), // Center
      ), // Scaffold
    ); // MaterialApp
  }
}
```

```dart
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        title: Text(widget.title),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ), // Text
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headlineMedium,
            ), // Text
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ),  // FloatingActionButton
    ); // Scaffold
  }
}
```

Using an AppBar, a Column and a Floating Action Button

The two apps on Android emulator





Looking at some comments in the „Application“

We learned about **setState** in “08 Stateful widgets.pdf”

```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++;
    });
  }
```

_incrementCounter is used in the Floating Action Button:

```dart
floatingActionButton: FloatingActionButton(
  onPressed: _incrementCounter,
  tooltip: 'Increment',
  child: const Icon(Icons.add),
),  // FloatingActionButton
```

Comments in MyApp

```dart
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 purple 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,
      ), // ThemeData
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    ); // MaterialApp
  }
}
```

This is only correct when debugging on Android, not in Chrome.

We learned about **ColorScheme** in “09 ThemeData and ColorScheme.pdf”

Comments in MyHomePage

This is an example, how parameters (here title) are defined in the c-tor of a stateful widget and then used in its state:

```dart
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Flutter Demo',
    theme: ThemeData(
      colorScheme: ColorScheme.fromSeed(seedColor: Colors.red),
      useMaterial3: true,
    ), // ThemeData
    home: const MyHomePage(title: 'Flutter Demo Home Page'),
  ); // MaterialApp
}
```

```dart
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 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
appBar: AppBar(
  // TRY THIS: Try changing the color here to a specific color (to
  // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
  // change color while the other colors stay the same.
  backgroundColor: Theme.of(context).colorScheme.inversePrimary,
  // 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),
```

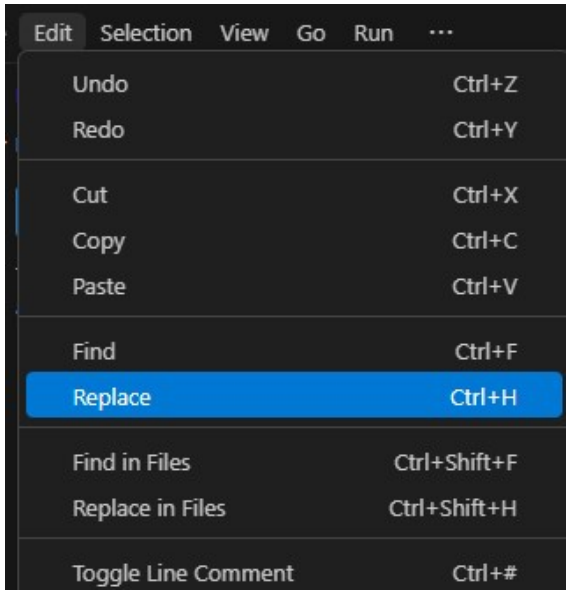
Comments in the build method of _MyHomePageState

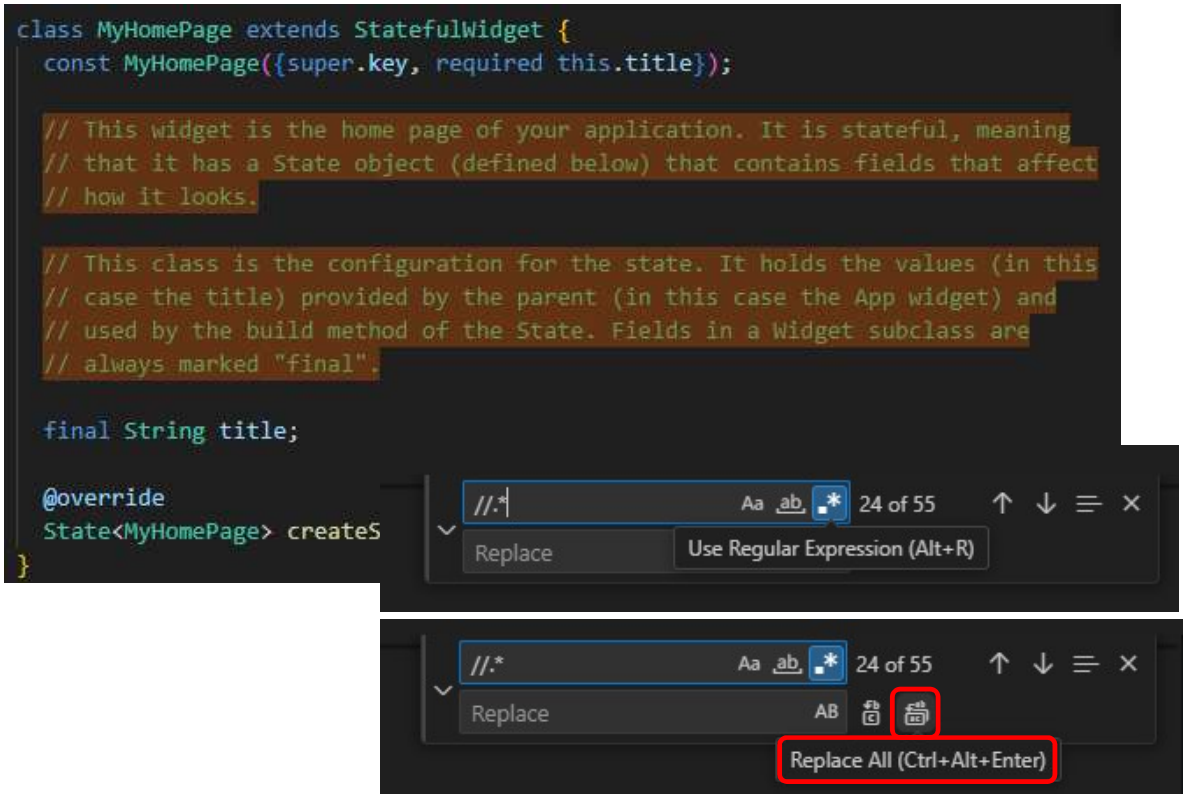
We learned about **Center** and **Column** widget very early in "03 Hello World app with texts and buttons.pdf"

New here: use predefined TextStyles from the Theme.

```
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.
    //
    // 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).
    //
    // TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
    // action in the IDE, or press "p" in the console), to see the
    // wireframe for each widget.
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      const Text(
        'You have pushed the button this many times:',
      ), // Text
      Text(
        '$_counter',
        style: Theme.of(context).textTheme.headlineMedium,
      ), // Text
    ], // <Widget>[]
  ), // Column
), // Center
```

Get rid of all comments in the „Application“





“.*” in a regular expression means: any character (“.”) any number of times (“*”)

Afterwards format the document with Shift+Alt+F to get rid of the empty lines.