Developing with Flutter

With Edd Gachira

Let’s create our first app

Prerequisites

- Make sure you can run ***flutter doctor*** in your cmd or terminal
- Open your VScode and make sure you have the flutter and dart extension installed

Creating your first flutter app

1. Open the Command Palette (Ctrl + Shift + P (Cmd + Shift + P on macOS)).
2. Select the Flutter: New Project command and press Enter .
3. Select Application and press Enter .
4. Select a Project location.
5. Enter your desired Project name.

Another way is to run on your cmd

flutter create my_app *(my_app being the name of your app)*

Let’s breakdown our app and file & folder structure

Folder structure

File structure

main.dart

Basic Widgets

These are the bundle of widgets that are absolutely necessary for the development of any flutter application.

Flutter Scaffold

The Scaffold widget is the base of the screen for a single page. It is used to implement the basic functional layout structure of an app. You can easily implement functional widgets like AppBar, FloatingActionButton, ButtonNavigationBar, Drawer, and many more widgets on the app using the Scaffold widget.

You can easily build an app using Scaffold and implement basic components with very less code, it can allow you to put all the material components to give look and feel to your app.

Constructors of Scaffold() widget:

AppBar

App bar is a horizontal bar that is displayed at the top of the screen. This is one of the main components of Scaffold widget. The app bar includes the toolbar icons, title of screen, quick action buttons.

To insert the app bar into your app, you need scaffold() widget:

Constructors of AppBar() widget:

Layout Widgets

This bundle of widgets helps in placing the other widgets on the screen as needed.

We know that flutter assume everything as a widget. So the image, icon, text, and even the layout of your app are all widgets. Here, some of the things you do not see on your app UI, such as rows, columns, and grids that arrange, constrain, and align the visible widgets are also the widgets.

Flutter allows us to create a layout by composing multiple widgets to build more complex widgets.

We can categories the layout widget into two types:

Single Child Widgets

The single child layout widget is a type of widget, which can have only one child widget inside the parent layout widget. These widgets can also contain special layout functionality. Flutter provides us many single child widgets to make the app UI attractive. If we use these widgets appropriately, it can save our time and makes the app code more readable. The list of different types of single child widgets are:

Container: It is the most popular layout widget that provides customizable options for painting, positioning, and sizing of widgets.

```
Center(
  child: Container(
    margin: const EdgeInsets.all(15.0),
    color: Colors.blue,
    width: 42.0,
    height: 42.0,
  ),
)
```

Padding: It is a widget that is used to arrange its child widget by the given padding. It contains EdgeInsets and EdgeInsets.fromLTRB for the desired side where you want to provide padding.

```
const Greetings(
  child: Padding(
    padding: EdgeInsets.all(14.0),
    child: Text('Hello JavaTpoint!'),
  ),
)
```

Center: This widget allows you to center the child widget within itself.

Align: It is a widget, which aligns its child widget within itself and sizes it based on the child's size. It provides more control to place the child widget in the exact position where you need it.

```
Center(
  child: Container(
    height: 110.0,
    width: 110.0,
    color: Colors.blue,
    child: Align(
      alignment: Alignment.topLeft,
      child: FlutterLogo(
        size: 50,
      ),
    ),
  ),
)
```

Multiple Child widgets

The multiple child widgets are a type of widget, which contains more than one child widget, and the layout of these widgets are unique. For example, Row widget laying out of its child widget in a horizontal direction, and Column widget laying out of its child widget in a vertical direction. If we combine the Row and Column widget, then it can build any level of the complex widget.

Here, we are going to learn different types of multiple child widgets:

Row: It allows to arrange its child widgets in a horizontal direction.

```
Center(
   child: Container(
    alignment: Alignment.center,
    color: Colors.white,
    child: Row(
     children: <Widget>[
      Expanded(
       child: Text('Peter', textAlign: TextAlign.center),
      ),
      Expanded(
       child: Text('John', textAlign: TextAlign.center ),

      ),
      Expanded(
       child: FittedBox(
        fit: BoxFit.contain, // otherwise the logo will be tiny
        child: const FlutterLogo(),
       ),
      ),
     ],
    ),
   ),
  )
```

Column: It allows to arrange its child widgets in a vertical direction.

ListView: It is the most popular scrolling widget that allows us to arrange its child widgets one after another in scroll direction.

GridView: It allows us to arrange its child widgets as a scrollable, 2D array of widgets. It consists of a repeated pattern of cells arrayed in a horizontal and vertical layout.

Expanded: It allows to make the children of a Row and Column widget to occupy the maximum possible area.

Table: It is a widget that allows us to arrange its children in a table based widget.

Flow: It allows us to implements the flow-based widget.

Stack: It is an essential widget, which is mainly used for overlapping several children widgets. It allows you to put up the multiple layers onto the screen. The following example helps to understand it.

Building Complex Layout

Let’s learn how to create a complex user interface using both single and multiple child layout widgets. The layout framework allows you to create a complex user interface layout by nesting the rows and columns inside of rows and columns.

Let us see an example of a complex user interface by creating the simple app.