Flutter Basic Application

Girma B.

material

- To start Flutter programming, you need first to import the Flutter package.
- Here, we have imported a Material package. This package allows you to create user interface according to the Material design guidelines specified by Android.

```dart
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primaryColor: Colors.teal,
      ),
      home: MyHomePage(),
      debugShowCheckedModeBanner: false,
    );
  }
}

class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Appbar"),
      ),
      body: const Center(
        child: Text("Home"),
      ),
    );
  }
}
```

runApp

- The second line is an entry point of the Flutter applications similar to the main method in other programming languages.
- It calls the runApp function and pass it an object of MyApp The primary purpose of this function is to attach the given widget to the screen.

Widget

- A widget used for creating UI in the Flutter framework. Here, the StatelessWidget does not maintain any state of the widget.
- MyApp extends StatelessWidget that overrides its build The build method is used for creating a part of the UI of the application.
- In this block, the build method uses MaterialApp, a widget to create the root level UI of the application and contains three properties - title, theme, and home.

Widget

- Title: It is the title of the Flutter application.
- Theme: It is the theme of the widget. By default, it set the blue as the overall color of the application.
- Home: It is the inner UI of the application, which sets another widget (MyHomePage) for the application.

MyHomePage

- the MyHomePage is similar to MyApp, except it will return the Scaffold.
- Scaffold widget is a top-level widget after the MaterialApp widget for creating the user interface.
- This widget contains two properties appBar and body. The appBar shows the header of the app, and body property shows the actual content of the application.
- Here, AppBar render the header of the application, Center widget is used to center the child widget, and Text is the final widget used to show the text content and displays in the center of the screen.

Flutter Architecture

- The Flutter architecture mainly comprises of four components.
  - Flutter Engine
  - Foundation Library
  - Widgets
  - Design Specific Widgets

Flutter Engine

- It is a portable runtime for high-quality mobile apps and primarily based on the C++ language.
- It implements Flutter core libraries that include animation and graphics, file and network I/O, plugin architecture, accessibility support, and a dart runtime for developing, compiling, and running Flutter applications.
- It takes Google's open-source graphics library, Skia, to render low-level graphics.

Foundation Library

- It contains all the required packages for the basic building blocks of writing a Flutter application.
- These libraries are written in Dart language.

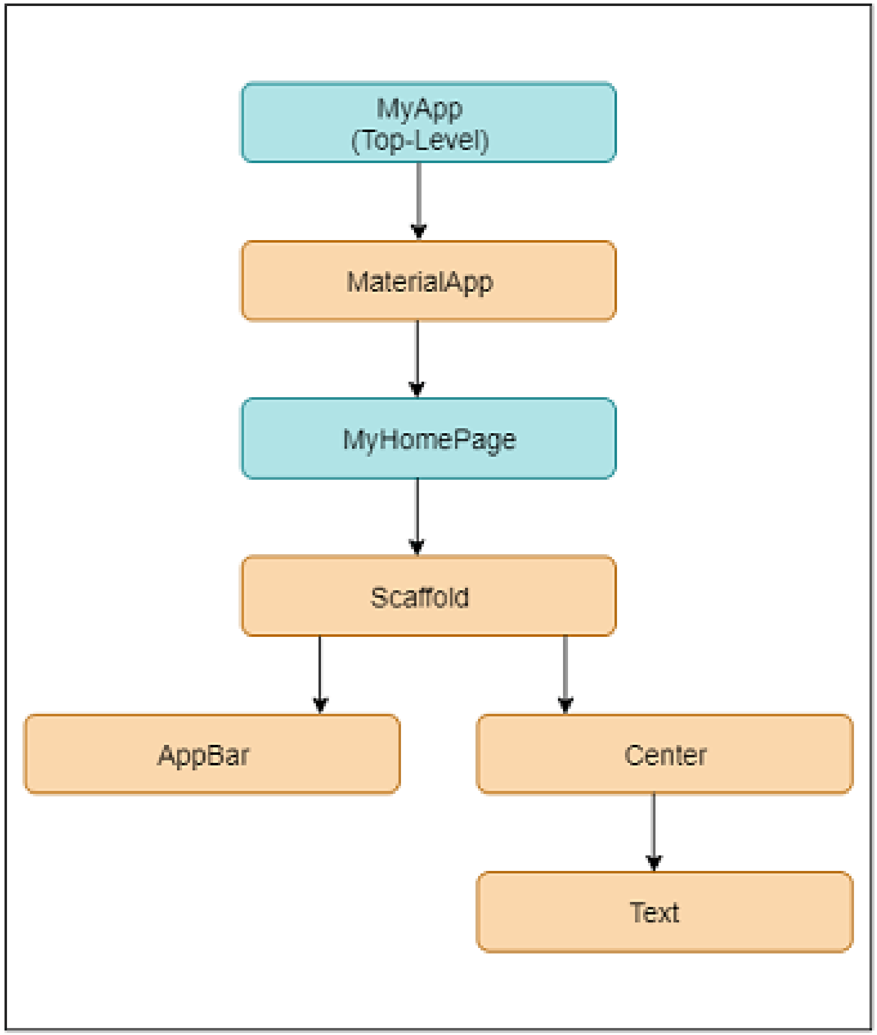
Widgets

- In Flutter, everything is a widget, which is the core concept of this framework.
- Widget in the Flutter is basically a user interface component that affects and controls the view and interface of the app.
- It represents an immutable description of part of the user interface and includes graphics, text, shapes, and animations that are created using widgets.
- The widgets are similar to the React components.

Widgets

- In Flutter, the application is itself a widget that contains many sub widgets.
- It means the app is the top-level widget, and its UI is build using one or more children widgets, which again includes sub child widgets.
- This feature helps you to create a complex user interface very easily.

Widgets


MyApp
(Top-Level)
MaterialApp
MyHomePage
Scaffold
AppBar
Center
Text

Design Specific Widgets

- The Flutter framework has two sets of widgets that conform to specific design languages.
- These are Material Design for Android application and Cupertino Style for IOS application.

Gestures

- It is a widget that provides interaction (how to listen for and respond to) in Flutter using GestureDetector.
- GestureDector is an invisible widget, which includes tapping, dragging, and scaling interaction of its child widget.
- We can also use other interactive features into the existing widgets by composing with the GestureDetector widget.

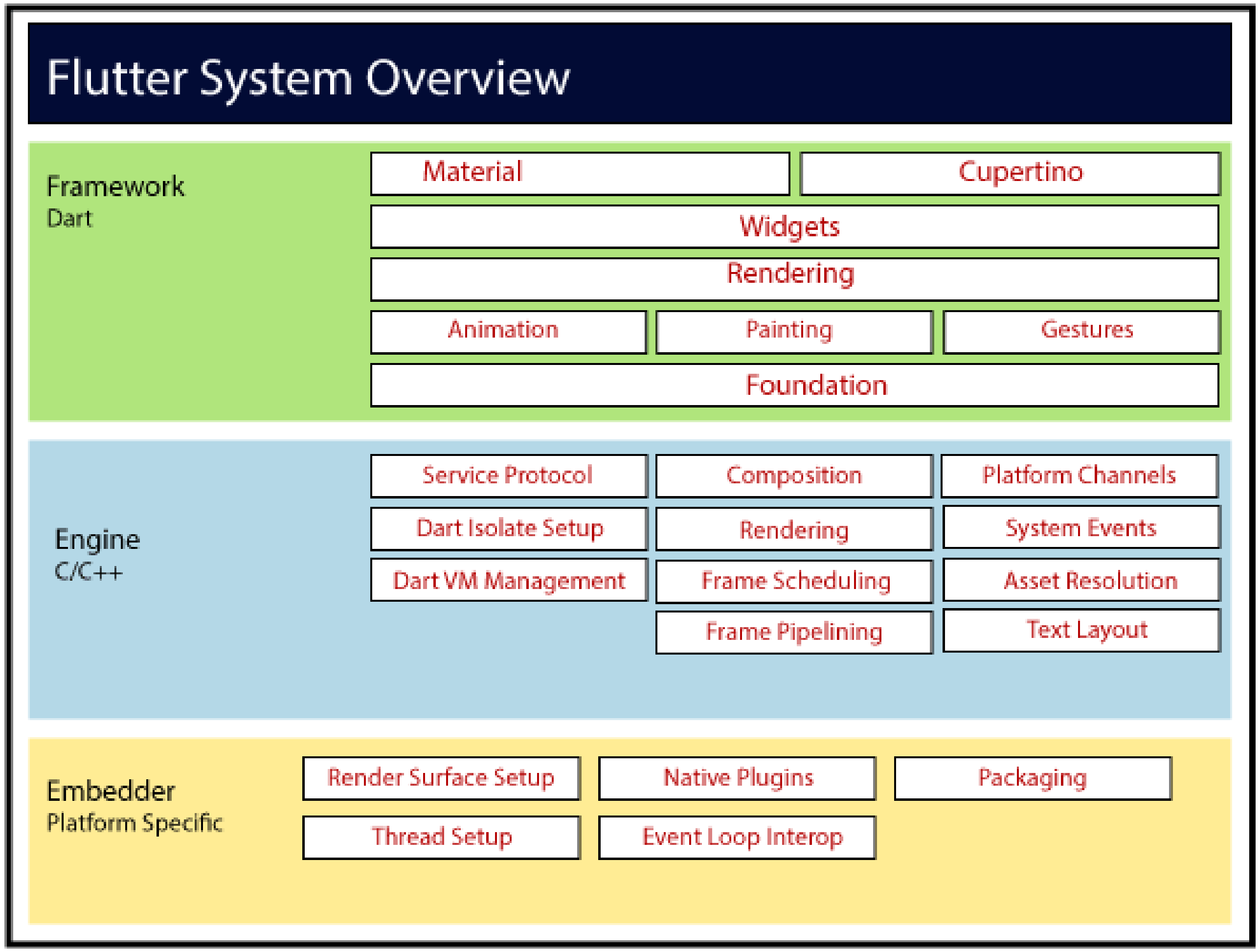
State Management

- Flutter widget maintains its state by using a special widget, StatefulWidget.
- It is always auto re-rendered whenever its internal state is changed.
- The re-rendering is optimized by calculating the distance between old and new widget UI and render only necessary things that are changes.

Layers

- Layers are an important concept of the Flutter framework, which are grouped into multiple categories in terms of complexity and arranged in the top-down approach.
- The topmost layer is the UI of the application, which is specific to the Android and iOS platforms. The second topmost layer contains all the Flutter native widgets.
- The next layer is the rendering layer, which renders everything in the Flutter app.
- Then, the layers go down to Gestures, foundation library, engine, and finally, core platform-specific code.

System Overview



Thank you