

Flutter –Cross Platform Ide for Mobile Applications

Vijayalakshmi S Katti¹ and Sowmya K²

¹⁻²Assistant Professor, Department of MCA, Nitte Meenakshi Institute of Technology, Bengaluru, Karnataka, India
Email: viju.s.katti@gmail.com, sowmyaraik.somi@gmail.com

Abstract—Flutter is a cross-platform mobile application development framework that allows users to create applications for both iOS and Android with a single codebase. Flutter applications are written in Dart programming language. For platform-specific code Flutter provides support for both Kotlin and Swift. Having a single code base, as with any other cross-platform mobile technology, can save an organization money and time by minimizing the need for developers to write and maintain two separate versions of an app (iOS and Android). From the consumer's viewpoint, this leads to lower product production costs.

Index Terms— Dart, Flutter,cross-platform frameworks, mobile applications.

I. INTRODUCTION

Choosing a mobile app is a time-consuming process. There are innumerable frameworks which offer good functionalities for developing mobile applications. You can write one piece of code and deliver it across many platforms using the cross-platform development framework (Android, iOS, and Desktop). Flutter is a user interface toolkit for creating natively built apps for mobile, web, and desktop.

It is open-source and free. Flutter is great for 2D mobile applications that run on both Android and iOS devices. We can also use it to create full-featured apps with camera, storage, geolocation, network, third-party SDKs, and Smartphone features. Flutter is unique from other frameworks in that it does not rely on Web View or the device's OEM widgets. Instead, it builds widgets with its own high-performance rendering engine. Most of its systems, such as animation, gesture, and widgets, are designed on the Dart programming language, which allows developers to simply read, alter, replace, or remove anything.

II. SCOPE

For a high-performance and amazing mobile application, Flutter comes with gorgeous and customizable widgets. It satisfies all custom requests and specifications. Dart has a big software package repository that you can use to expand the functionality of your application. Developers just need to write a single code base for both applications: Android and iOS. Flutter could be ported to other more platforms in the future. Testing is easier using Flutter. It is sufficient to develop automated tests for both systems once due to the shared code base. Because of its simplicity, Flutter is an excellent contender for quick development. It's even more powerful because of its customizability and extensibility. Developers have complete control over the framework with Flutter.

III. LITERATURE REVIEW

With several platforms and associated software development kits, mobile application development is becoming increasingly difficult^[1]. Modern smartphones use a wide range of mobile operating systems (OS), such as Google's Android, Apple's iOS, Microsoft's Windows Phone, and so on^[2]. The examination of these WORA (Write Once, Run Anywhere) mobile App tools is done on the basis of different approaches, i.e., Native apps, Web apps, Hybrid apps, Interpreted apps and Widget-based apps^[6]. Because there are multiple mobile platforms (iOS, Android, and so on), each with its own SDK (Software Development Kit) tools and development capabilities, application development becomes more involved and costly^[3]. Cross-platform development solutions aim to facilitate production of mobile applications for multiple platforms from a single codebase^[7]. Cross-platform frameworks that show resemblance to React Native and Flutter, are discussed and implemented by various companies numerous times formerly^[8]. Due to a lack of development resources, developers are expected to accept only a few platforms and devices^[4]. Currently existing cross-platform frameworks take numerous techniques to deploying the completed application, each with its own set of advantages and disadvantages^[5].

IV. WORKING ANALYSIS

Everything in Flutter framework is widget, which is the core. Widgets are mainly used by the developers to develop the user interface of the mobile applications. The root of all the widgets is programme; the user interface of this root widget is created by one or more child process (widgets), which then fork other widgets as required by the process. The capacity to create user interfaces of any complexity is aided by the virtue of compos-ability. Maximum File size supported by the flutter is 20mb. The hello world program's widget hierarchy (described below) is, for example, as illustrated in the diagram.

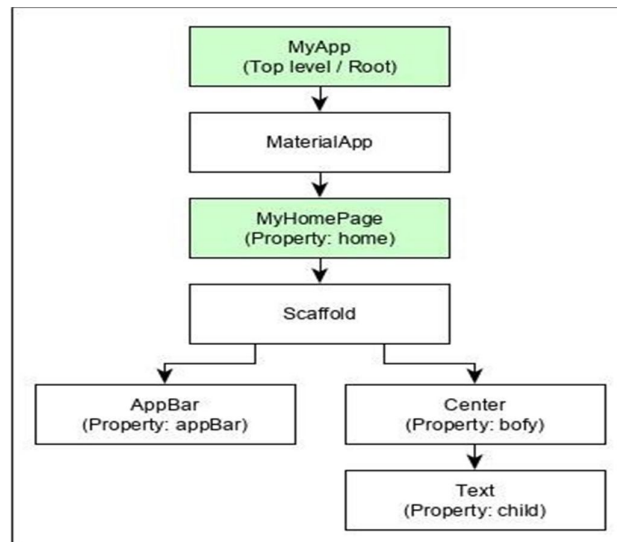


Figure 4.1 depicts the basic steps of creating the Hello world Program using the application

The following points are noteworthy in this regard:

- MyApp is a user-created widget made with MaterialApp, a native Flutter widget.
- The properties in the MaterialApp provides the functionality for creation of user interface which is a widget: MyHomePage.
- Scaffold is the built in app of Flutter framework for providing the properties like AppBar and Body widgets.
- The *body* specifies the primary user interface, while *AppBar* specifies the header.
- The Header UI is made with a flutter native widget, while the AppBar and Body UI are made with a Center widget.
- The Center widget, which is formed with the Text widget, has a Child attribute that corresponds to the actual content.

A. Gestures

GestureDetector, an unique widget in Flutter, allows for interactivity. GestureDetector is an invisible widget that may record user events with its child widget, such as tapping, dragging, and so on. GestureDetector is used by several native Flutter widgets to provide interaction. By mixing the existing widget with the GestureDetector widget, we can add an interactive element to it.

B. State Management

Flutter provides the state management with is StatefulWidget and setState() method. The state derived from these widgets can be used to provide the state management across the different parts of the application and the user session. The fast development feature of flutter will rebuild the user interface from the scratch with its stateful hot reload.

C. Layers

The Flutter framework is organised into broad Layers where each layers are independent libraries which offer the dependencies for the underlying layers. The layers are optional and replaceable, cannot access the below layers. The platform specific Embedder is the entry point, which coordinates with the OS for providing the services. The flutter engine is written in c/c++ language, which can used using the dart: ui; for providing the library features of engine to the layers. The foundation layer provides the basic foundation services as, painting, animation and gestures. Rendering layers offers the conversion of widgets to pixels. Widgets layers offer the reactive programming, classes for creating the user interface. To design the Material or iOS languages the libraries from the material and Cupertino, which offer the controls from widgets, are used.

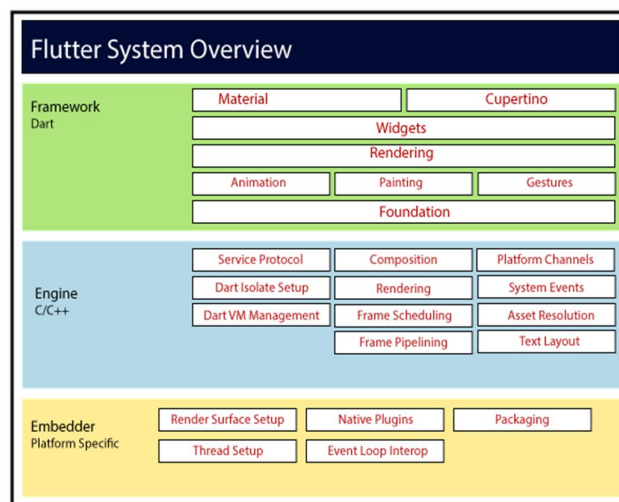


Figure 4.2 depicts the architecture of Flutter

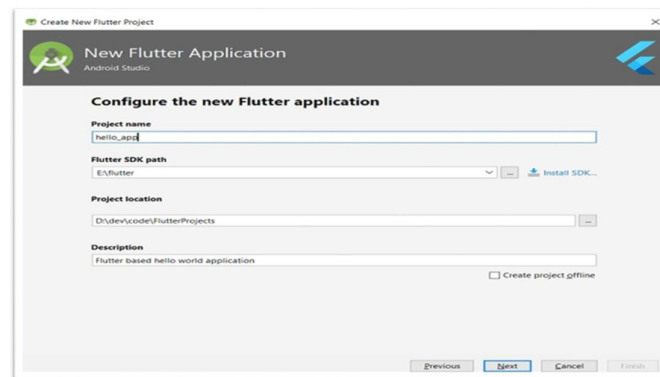


Figure 4.3 depicts the configuration of new flutter app



Figure 4.4 depicts the basic home page

V. RESULTS

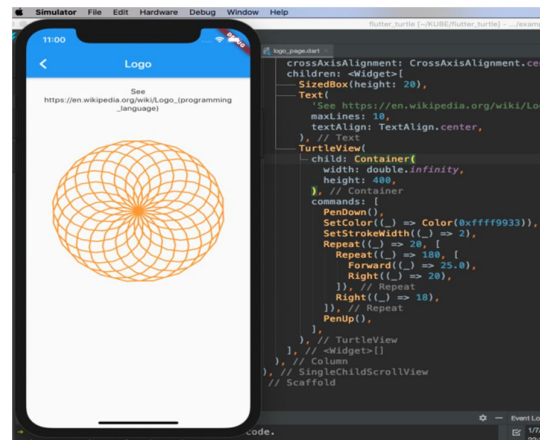


Figure 5.1 description- flutter_turtle is a simple implementation of turtle graphics for Flutter. It simply uses a custom painter to draw graphics by a series of Logo-like commands

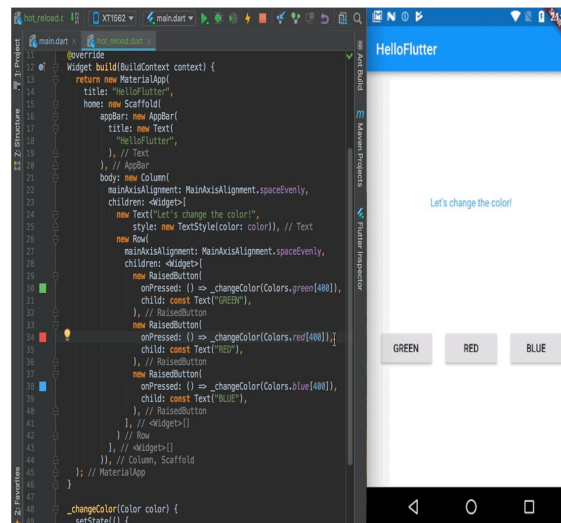


Figure 5.2 Text widget supports Text Styling which supports color property. We can pass here color code in different formats like Hex Color code, Inbuilt color constants and RGB color codes

VI. CONCLUSION

Flutter is the good framework for creating the mobile applications, with its massive collection of widgets and the material for designing the user friendly interface with ease. The stateful widget of flutter applications enhances the development of application and allows the developers to see the changes in code as used for development without restarting the application and the loss of state. The framework offers the platform-independent feature for application development, high performance and user friendly interface for developing both Android and iOS applications.

REFERENCES

- [1] Dalmaso, I., Datta, S. K., Bonnet, C., & Nikaein, N. (2013, July). Survey, comparison and evaluation of cross platform mobile application development tools. In *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)* (pp. 323-328). IEEE.
- [2] Cha, S. H., & Yun, Y. (2013). Smartphone Application Development using HTML5-based Cross-Platform Framework.
- [3] Latif, M., Lakhrissi, Y., & Es-Sbai, N. (2016, March). Cross platform approach for mobile application development: A survey. In *2016 International Conference on Information Technology for Organizations Development (IT4OD)* (pp. 1-5). IEEE.
- [4] Dhillon, S., & Mahmoud, Q. H. (2015). An evaluation framework for cross-platform mobile application development tools. *Software: Practice and Experience*, 45(10), 1331-1357.
- [5] Ciman, M., & Gaggi, O. (2017). An empirical analysis of energy consumption of cross-platform frameworks for mobile development. *Pervasive and Mobile Computing*, 39, 214-230.
- [6] Shah, K., Sinha, H., & Mishra, P. (2019, March). Analysis of Cross-Platform Mobile App Development Tools. In *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)* (pp. 1-7). IEEE.
- [7] Dagne, L. (2019). Flutter for cross-platform App and SDK development.
- [8] Tashildar, A., Shah, N., Gala, R., Giri, T., & Chavhan, P. (2020). Application development using flutter. *International Research Journal of Modernization in Engineering Technology and Science*, 2(08), 1262-1266.