Flutter Widgets

Tushar B. Kute,
http://tusharkute.com

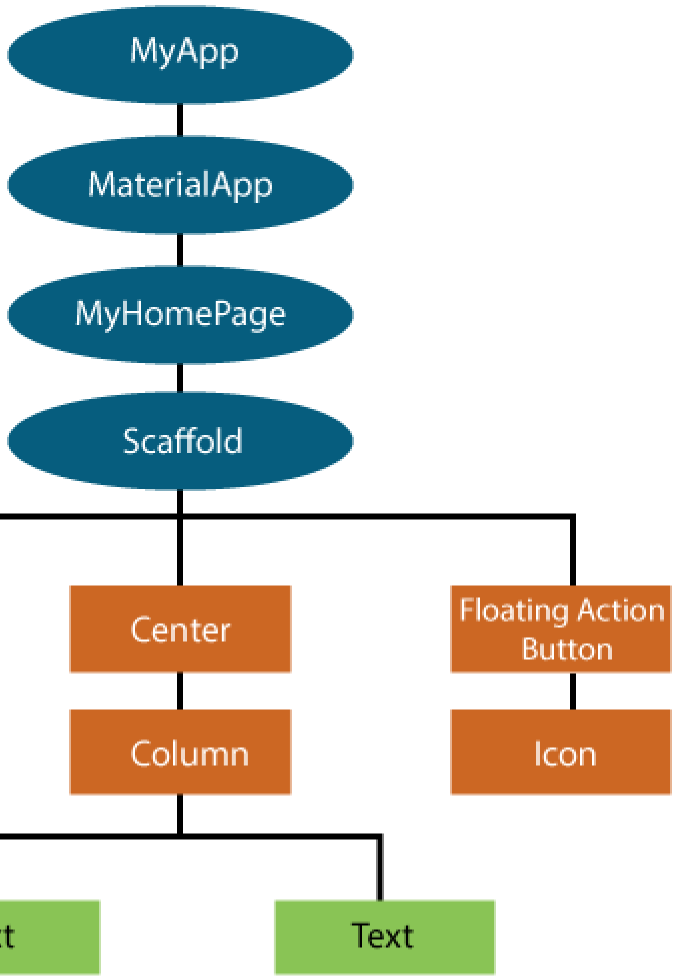
Widget

- Whenever you are going to code for building anything in Flutter, it will be inside a widget.
- The central purpose is to build the app out of widgets. It describes how your app view should look like with their current configuration and state.
- When you made any alteration in the code, the widget rebuilds its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in UI of the app.

Widget

- Widgets are nested with each other to build the app. It means the root of your app is itself a widget, and all the way down is a widget also.
- For example, a widget can display something, can define design, can handle interaction, etc.

Widget Tree


MyApp
MaterialApp
MyHomePage
Scaffold
AppBar
Center
Floating Action Button
Text
Column
Icon
Text
Text

Widget Creation

- We can create the Flutter widget like this:

```
Class ImageWidget extends StatelessWidget {
        // Class Stuff
}
```

Stateful and Stateless

- A widget is either stateful or stateless. If a widget can change—when a user interacts with it, for example—it's stateful.
- A stateless widget never changes. Icon, IconButton, and Text are examples of stateless widgets. Stateless widgets subclass StatelessWidget.
- A stateful widget is dynamic: for example, it can change its appearance in response to events triggered by user interactions or when it receives data.
- Checkbox, Radio, Slider, InkWell, Form, and TextField are examples of stateful widgets. Stateful widgets subclass StatefulWidget.

Widget - Types

- We can split the Flutter widget into two categories:
  - Visible (Output and Input)
  - Invisible (Layout and Control)

Visible Widget

- The visible widgets are related to the user input and output data. Some of the important types of this widget are:
- Text
  - A Text widget holds some text to display on the screen.
  - We can align the text widget by using textAlign property, and style property allow the customization of Text that includes font, font weight, font style, letter spacing, color, and many more. We can use it as like below code snippets.

Visible Widget

```
new Text(
  'Hello, Skillologies!',
  textAlign: TextAlign.center,
  style: new TextStyle(fontWeight: FontWeight.bold),
  )
```

Visible Widget

```
//FlatButton Example
new FlatButton(
  child: Text("Click here"),
  onPressed: () {
    // Do something here
  },
),
```

Visible Widget

```
//RaisedButton Example
new RaisedButton(
  child: Text("Click here"),
  elevation: 5.0,
  onPressed: () {
    // Do something here
  },
),
```

Image

This widget holds the image which can fetch it from multiple sources like from the asset folder or directly from the URL. It provides many constructors for loading image, which are given below:

- Image: It is a generic image loader, which is used by ImageProvider.
- asset: It load image from your project asset folder.
- file: It loads images from the system folder.
- memory: It load image from memory.
- network: It loads images from the network.

Visible Widget

- The visible widgets are related to the user input and output data. Some of the important types of this widget are:
- Text
  - A Text widget holds some text to display on the screen.
  - We can align the text widget by using textAlign property, and style property allow the customization of Text that includes font, font weight, font style, letter spacing, color, and many more. We can use it as like below code snippets.

Icon

- This widget acts as a container for storing the Icon in the Flutter. The following code explains it more clearly.

```
new Icon(
  Icons.add,
  size: 34.0,
)
```

Invisible Widget

- The invisible widgets are related to the layout and control of widgets. It provides controlling how the widgets actually behave and how they will look onto the screen. Some of the important types of these widgets are:
- Column
  - A column widget is a type of widget that arranges all its children's widgets in a vertical alignment.
  - It provides spacing between the widgets by using the mainAxisAlignment and crossAxisAlignment properties.
  - In these properties, the main axis is the vertical axis, and the cross axis is the horizontal axis.

Invisible Widget

- The below code snippets construct two widget elements vertically.

```
new Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    new Text(
      "VegElement",
    ),
    new Text(
      "Non-vegElement"
    ),
  ],
),
```

Invisible Widget

- The row widget is similar to the column widget, but it constructs a widget horizontally rather than vertically.
- Here, the main axis is the horizontal axis, and the cross axis is the vertical axis.
- Example
  - The below code snippets construct two widget elements horizontally.

Invisible Widget

```
new Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: <Widget>[
    new Text(
      "VegElement",
    ),
    new Text(
      "Non-vegElement"
    ),
  ],
),
```

Invisible Widget

- Center: This widget is used to center the child widget, which comes inside it. All the previous examples contain inside the center widget.

```
Center(
  child: new clumn(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: <Widget>[
      new Text(
        "VegElement",
      ),
      new Text(
        "Non-vegElement"
      ),
    ],
  ),
),
```

Invisible Widget

- This widget wraps other widgets to give them padding in specified directions. You can also provide padding in all directions. We can understand it from the below example that gives the text widget padding of 6.0 in all directions.

  Example

```
Padding(
  padding: const EdgeInsets.all(6.0),
  child: new Text(
    "Element 1",
  ),
),
```

Invisible Widget

- Scaffold
  - This widget provides a framework that allows you to add common material design elements like AppBar, Floating Action Buttons, Drawers, etc.
- Stack
  - It is an essential widget, which is mainly used for overlapping a widget, such as a button on a background gradient.

StatefulWidget

- A StatefulWidget has state information. It contains mainly two classes: the state object and the widget. It is dynamic because it can change the inner data during the widget lifetime. This widget does not have a build() method. It has createState() method, which returns a class that extends the Flutters State Class. The examples of the StatefulWidget are Checkbox, Radio, Slider, InkWell, Form, and TextField.

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.

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com