Flutter:Single/Multi-Child Widgets

Prashanth B S[1]

[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology,
Yelahanka - 560064, Bengaluru

1 Layout Widgets

While designing apps using flutter, many inbuilt widgets from Flutter SDKs are used. Based on the number of childs a widget can possess, they are broadly classified as,

1. *Single Child Widgets* - These widgets have a single child property often seen as "child: ...".
2. *Multi Child Widgets*- These widgets can have more then one child, children in particular. often seen as "children[]" property

Some of the examples of the Single child widgets are container, padding, Center, Align, SizedBox Widget, etc. While for the Multi child widgets the most commonly used widgets are row and column widgets which can hold multiple widgets in bunch of rows and columns. The section aims at exploring few commonly used Layout Widgets, i.e, Single and Multi child Widgets.

2 Single-Child Widgets

Few commonly used Single-Child Widgets are discussed in this section. The focus is on Container, padding and Center Widgets.These widgets can house only one child widget.

1. *Container Widget*: Container Widgets houses one Widget, it provides two attributes for customization, they are sizeheight and width, painting & positioning[1]. The following code snippet shows the same,

```
body: Container(
    height: 200.0, // 200 logical pixel - depends on Device
    width: double.infinity,//expand to the width of screen
    color: Colors.blue, // Color attribute for the container
    child: Center( // Container ->child(Center->child(Text))
         child: Text(
         'Hello World',
         style: TextStyle(color: Colors.white, fontSize: 30),
            ),
        ),
    ),
```

2. *Center Widget*: Center Widget positions its child widget at the center of the screen.

[1]https://chamithchathuka.medium.com/flutter-layout-widgets-b137758d9d43

```
body: Center(
        child: Text(
        'Hello World',
        style: TextStyle(color: Colors.white, fontSize: 30),
        ),
)
```

3. *Align Widget*: Align Widgets places its child w.r.t to its parent using alignment property whose values can be topCenter, topRight, topLeft, center, bottomCenter, bottomRight, bottomLeft using the Align Object. The following code snippet has the parent Container which houses a child Aligment. The Align acts as a parent for another container.

```
body: Container(
    height: 200.0,
    width: double.infinity,
    color: Colors.blue, // Color attribute for the outer container
    child: Align(
    alignment: Alignment.bottomCenter, //places the child of align to bottomCenter
        of the outer container
    //change the topCenter, topRight, topLeft, center, bottomCenter, bottomRight,
        bottomLeft and observe the change
    child:Container( // inner container
    height: 20.0,
    width: 20.0,
    color: Colors.white, // Color attribute for the inner container
    ),
  ),
),
```

4. *Padding Widget*: This widget helps to add padding to its child widget. It provides a property called as padding which is set using an Object of *EdgeInset* Class. Some of the the options for padding using EdgeInset class are,

   (a) *EdgeInset.all(double value)*: Sets the padding from all direction using the value in terms of pixels

   (b) *EdgeInset.fromLTRB(double left, double top, double right, double bottom)*: sets the padding by adjusting the arguments of left, top, right and bottom

   (c) *EdgeInset.symmetric(double vertical=0.0, double horizontal=0.0)*: Sets the horizontal padding(left, right) and vertical padding(top,bottom)

   (d) *EdgeInset.only(double left, double top, double right, double bottom)*: same as fromLTRB

   The following code snippet demonstrate the usage of Padding,

```
Padding(
    padding: const EdgeInsets.all(20.0), // puts 20 dp padding from all sides
    // try setting the EdgeInset to other functions as well
    child: Container(
        height: 200.0,
        width: double.infinity,
        color: Colors.blue,
        ),
)
```

3 Multi-child Widgets

Multi child widgets are the widgets which supports more than one child widgets. Some of the commonly used multi-child widgets are, ScrollView, Row & Column Widgets. This section presents the discussion on Row and Column Widgets in particular.

3.1 Row Widget

A row widget places his children horizontally in a form of row. The row widget supports children[child1_widget, child2_widget,...] property to let the user define childrens for the row. The horizontal axis of the row is called *MainAxis* and the perpendicular axis is called as *CrossAxis*. These properties can be set using MainAxisAlignment & CrossAxisAlignment property respectively. A row can have multiple row/column widgets as its childrens.

3.2 Column Widget

A column widget places his children vertically in a form of column. The column widget supports children[child1_widget, child2_widget,...] property to let the user define childrens for the column. The horizontal axis of the row is called *CrossAxis* and the perpendicular axis is called as *MainAxis* for the column. These properties can be set using MainAxisAlignment & CrossAxisAlignment property respectively.A column can have multiple row/column widgets as its childrens.

4 Exercise-I

Build a flutter app to demonstrate the usage of Row Widget and Experiment with its MainAxis and CrossAxis properties respectively.
The Steps followed are as follows,

1. Create a MaterialApp and add Custom RowFun() as Homepage of the flutter app as shown below.

```
void main() {
  runApp(MaterialApp(
    home: RowApp(),
  ));
}
```

2. Make the RowApp inherit the traits of Stateless widget to support hot-reload features.

```
class RowApp extends StatelessWidget { // Our RowApp page inherits the traits of the
    Stateless Widget
  const RowApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(); // here we build the Widget Tree
   }
}
```

3. Add the Row Widget to body of the App. Set the mainAxisAlignment as MainAxisAlignment.spaceEvenly(start,end,center, spaceAround, strech) and crossAxisAlignment as CrossAxisAlignment.start(end,stretch,baseline).

```
body:Row(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.start,
        children:[ ],
)
```

4. Make three containers as a children of Row, and specify the children inside children:[ ]. This is demonstrated as shown in the below code snippet.

```
Container(
     height: 100,
     width: 100,
     color: Colors.blue, // change the color to red and green for other two
         containers
     child: Center(
     child: Text(
            'First Child'
      ),
   ),
),
// code for container 2
// code for container 3
```

5 Exercise-II

Build a flutter app to demonstrate the usage of Column Widget and Experiment with its MainAxis and CrossAxis properties respectively.
The Steps followed are as follows,

1. Create a MaterialApp and add Custom ColumnApp() as Homepage of the flutter app as shown below.

```
void main() {
  runApp(MaterialApp(
    home: ColumnApp(),
  ));
}
```

2. Make the ColumnApp inherit the traits of Stateless widget to support hot-reload features.

```
class ColumnApp extends StatelessWidget { // Our ColumnApp page inherits the traits
    of the Stateless Widget
  const ColumnApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(); // here we build the Widget Tree
   }
}
```

3. Add the Column Widget to body of the App. Set the mainAxisAlignment as MainAxisAlignment.spaceEvenly(start,end,center, spaceAround, strech) and crossAxisAlignment as CrossAxisAlignment.start(end,stretch,baseline).

```
body:Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.start,
        children:[ ],
)
```

4. Make three containers as a children of Column, and specify the children inside children:[ ]. This is demonstrated as shown in the below code snippet.

```
Container(
     height: 100,
     width: 100,
     color: Colors.blue, // change the color to red and green for other two
         containers
     child: Center(
     child: Text(
           'First Child'
      ),
   ),
),
// code for container 2
// code for container 3
```

6 Exercise-III

Build a flutter app to demonstrate the usage of Row & Column Widget and Experiment with its MainAxis and CrossAxis properties respectively for the design shown in figure 1.

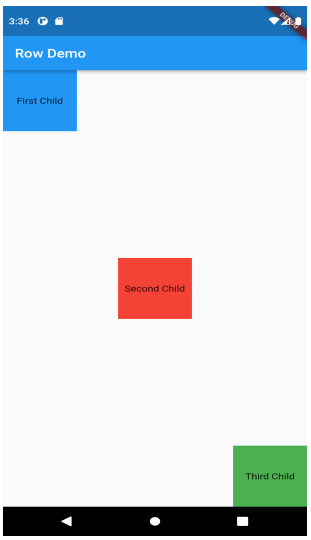
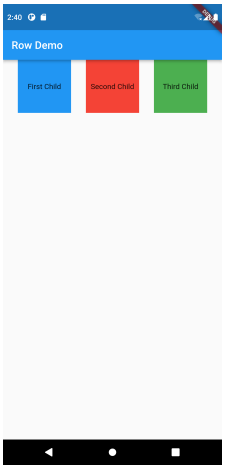
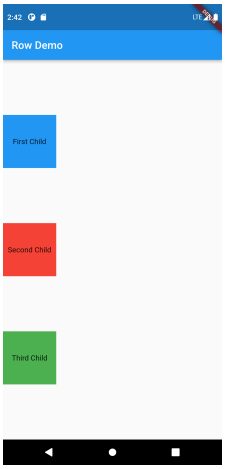


Figure 1: TO-DO Exercise



(a) Row Widget Demo



(b) Column Widget Demo

Figure 2: Row & Column Widget Demo

7 Results

The results of Row and Column Widgets are shown in below. The figures 2a shows the output of Row Widget and 2b shows the Column Widget.