Stateful Widgets: (Refer App: FluffyBanana)

Its time now to go dynamic. We’ve seen enough of the different types of layout widgets, text styles, images, container and everything else of that sort. Using whatever we learnt till now, we created a simple app which presents a layout similar to that of a profile page or an ‘about me’ page of any app which requires us to create an account. A brief summary of the new things we learnt are as follows:

* In order to quickly separate UI elements (a sort of quick padding if you may), we can use the SizedBox() widget. This is a transparent box, with 2 properties: width and height. We can adjust them to our specification, in order to add space between widgets
* Flutter provides us with a readymade material design widget to display the profile picture of the user in a circle (a thing which we see in many apps). This widget is called ‘CircleAvatar’. Here we can specify the image to be inserted inside the circle, as well as the radius of the circle.
* Flutter also provides us with a material design widget called ‘Divider’. This is a coloured line with spacing around it. It is used as a horizontal rule to separate different elements. This line has a property called ‘height’ which is the total no. of pixels below as well as above the line.
* The method to add a toast message to flutter is slightly different. Because flutter is optimised for material design, material design prefers to use snackbars to show quick notifications instead of toasts. But, still if we do want to display toasts, we use an external library whose link is [here](https://pub.dev/packages/fluttertoast) and the instructions are [here](https://stackoverflow.com/questions/45948168/how-to-create-toast-in-flutter).
* To change the launcher icon, the process is unfortunately complicated (but not that much). In the pubspec.yaml file, we need to add a dependency to our app called ‘flutter\_launcher\_icons’ which automatically creates the icons for android as well as iOS and then adds them to their respective directories. Instructions to use are [here](https://stackoverflow.com/questions/43928702/how-to-change-the-application-launcher-icon-on-flutter) and if you need it in a video format to follow along, find it [here](https://www.youtube.com/watch?v=mysyraPhQ5k) and [here](https://www.youtube.com/watch?v=hpQenyqxTmw). Generate a launcher icon [here](https://romannurik.github.io/AndroidAssetStudio/).

A stateful widget as stated earlier, is the one whose content changes dynamically. In order to create a stateful widget, android studio provides us with a handy little code snippet which is activated when we type in the words ‘stful’. Let’s name our widget as ‘Test’.

**class** Test **extends** StatefulWidget {  
 @override  
 \_TestState createState() => \_TestState();  
}  
  
**class** \_TestState **extends** State<Test> {  
 @override  
 Widget build(BuildContext context) {  
 **return** Container();  
 }  
}

The first thing we notice is that this widget consists of not one but TWO classes. The first class consists of a method called ‘createState()’, which returns an object of the \_TestState() class.

The second class is the \_TestState class. We call the object of this class as the ‘state object’. This is the class where we specify the widget tree of our dynamic app. This widget tree contains dynamic elements like variable names which hold varying quantities instead of static numbers. Whenever the values of the variables change, we say that the ‘state’ of our app has changed.

Now because the state of our app has changed, there must be some changes in the UI. In order to display these changes, we need to rebuild the widget tree. This job is performed by the ‘createState()’ function of the upper class.

Thus, whenever the app state changes, the createState() function recreates the widget tree, and repaints the UI. For this reason, it returns the state object as the output. The class containing the createState function is our widget. Thus, the name of our widget is Test().

We place all of the widget tree as the return parameter of the \_TestState class (similar to how we did with stateless widgets).

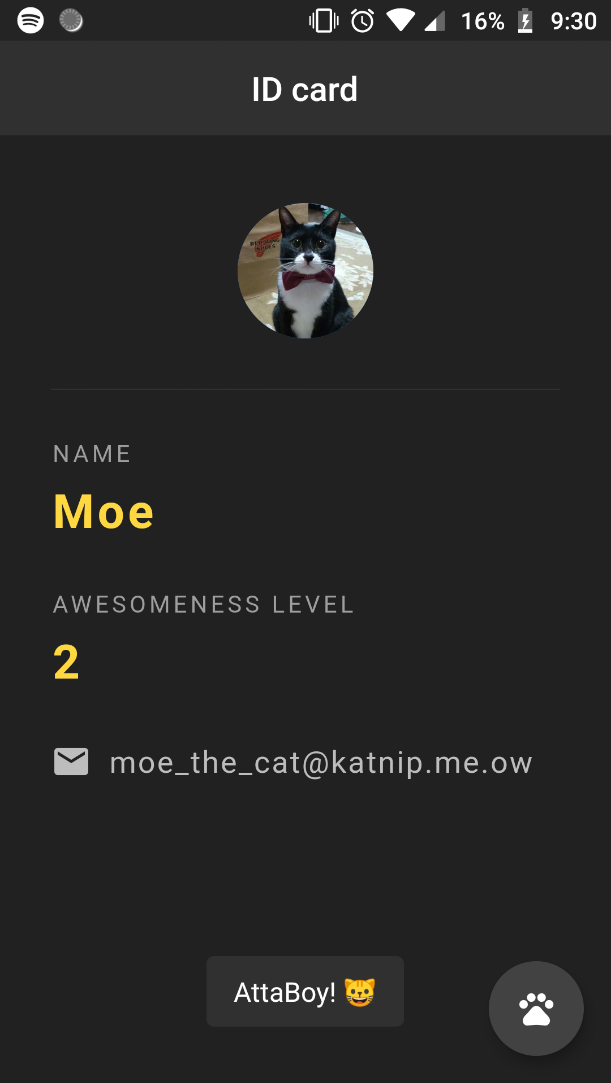
An important thing to remember is that we cannot directly change UI data in our app. We somehow need to tell flutter that it needs to repaint the screen after the data has been changed. For this reason, we use what is called a ‘SetState()’ function. This function takes a FUNCTION as the input. This function can be an anonymous function (a function without a name), kind of what we were specifying to the onClicked property of buttons. INSIDE this anonymous function, we specify what changes we want to make to the UI, and the SetState() function will automatically rebuild the widget tree with the updated values.

A handy trick to convert a stateless widget into a stateful one, is to hover the mouse over the ‘other options’ menu (the yellow lightbulb in android studio) and then choose ‘convert to stateful widget’.

The final app code looks like:

**import 'package:flutter/material.dart'**;  
**import 'package:fluttertoast/fluttertoast.dart'**;  
  
**void** main() => runApp(MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
 home: MainCard() ,  
));  
  
**class** MainCard **extends** StatefulWidget {  
 @override  
 \_MainCardState createState() => \_MainCardState();  
}  
  
**class** \_MainCardState **extends** State<MainCard> {  
  
 int **awesmlvl**=0;  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 backgroundColor: Colors.*grey*[900],  
 appBar: AppBar(  
 title: Text(**'ID card'**) ,  
 centerTitle: **true**,  
 backgroundColor: Colors.*grey*[850],  
 elevation: 0.0,  
 ),  
 floatingActionButton: FloatingActionButton(  
 onPressed: (){  
 setState(() {  
 **awesmlvl**=**awesmlvl**+1;  
 });  
 Fluttertoast.*showToast*(  
 msg: **"AttaBoy! 😺"**,  
 toastLength: Toast.**LENGTH\_SHORT**,  
 gravity: ToastGravity.**BOTTOM**,  
 timeInSecForIosWeb: 1,  
 backgroundColor: Colors.*grey*[850],  
 textColor: Colors.*white*,  
 fontSize: 16.0  
 );  
 },  
 child: Icon(  
 Icons.*pets*,  
 ),  
 backgroundColor: Colors.*grey*[800],  
 ),  
 body: Padding(  
 padding: EdgeInsets.fromLTRB(30.0, 40.0,30.0, 0.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**start**,  
 children: <Widget>[  
 Center(  
 child: CircleAvatar(  
 backgroundImage: AssetImage(**'assets/tuxedocat1.jpg'**),  
 radius: 40.0,  
 ),  
 ),  
 Divider(  
 height: 60.0,  
 color: Colors.*grey*[800],  
 ),  
 Text(  
 **'NAME'**,  
 style: TextStyle(  
 color: Colors.*grey*,  
 letterSpacing: 2.0,  
 ),  
 ),  
 SizedBox(height: 10.0),  
 Text(  
 **'Moe'**,  
 style: TextStyle(  
 color: Colors.*amberAccent*[200],  
 letterSpacing: 2.0,  
 fontSize: 28.0,  
 fontWeight: FontWeight.*bold* ),  
 ),  
 SizedBox(height: 30.0),  
 Text(  
 **'AWESOMENESS LEVEL'**,  
 style: TextStyle(  
 color: Colors.*grey*,  
 letterSpacing: 2.0,  
 ),  
 ),  
 SizedBox(height: 10.0),  
 Text(  
 **'**$**awesmlvl'**,  
 style: TextStyle(  
 color: Colors.*amberAccent*[200],  
 letterSpacing: 2.0,  
 fontSize: 28.0,  
 fontWeight: FontWeight.*bold* ),  
 ),  
 SizedBox(height:30.0),  
 Row(  
 children: <Widget>[  
 Icon(  
 Icons.*email*,  
 color: Colors.*grey*[400],  
 ),  
 SizedBox(width: 10.0),  
 Text(  
 **'moe\_the\_cat@katnip.me.ow'**,  
 style: TextStyle(  
 color: Colors.*grey*[400],  
 fontSize: 18.0,  
 letterSpacing: 1.0  
 ),  
 )  
 ],  
 )  
 ],  
 ),  
 ),  
 );  
 }  
}

The output looks like:



Lists of Data: (Refer app: Wordy)

Flutter provides us with a utility similar to listView in the android-java environment. For an example, let’s create a list of strings into our stateful widget.

List<String> **quotes**=[  
 **'If a black cat crosses your path, it signifies that it is going somewhere'**,  
 **'A cat is an example of sophistication minus civilisation'**,  
 **'When a cat closes its eyes, there is a good chance that it cannot see'**];

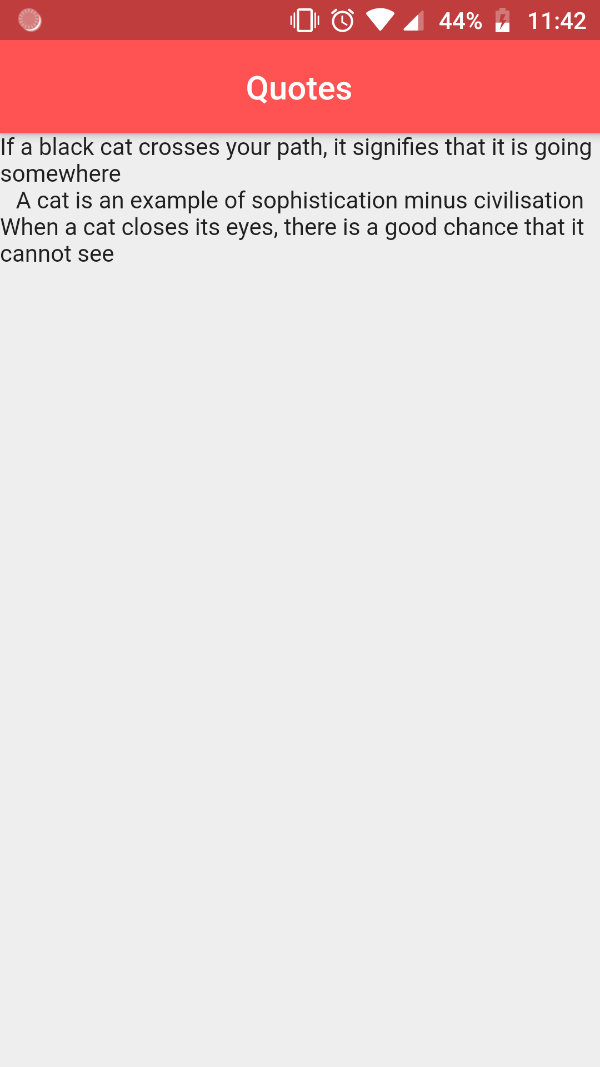
Now, we want to display these strings on the body of our app. We could easily create separate Text() widgets for each of the strings. But it would be tedious if we had a lot of strings in our list. Instead, we have a trick up our sleeve.

We’ll create a column widget, as we want to display the strings one below the other. In the children property, instead of specifying a list of widgets, we’ll use the ‘map’ function on our list of quotes. The map function in dart accepts another function as the parameter and returns an iterable. The map function has the ability to cycle through all the elements one by one in our quote list. Once we have access to one quote inside the list, the function inside ‘map’ will specify what exactly we want to do with that quote.

We can give any name to the parameter of the inside anonymous function. We also need not specify the datatype of the parameter. The map function automatically calls the inside function for every element in the List, and automatically passes the obtained element into the function. Thus, the name ‘quote’ is only for our reference.

Well, we need to put it inside a Text() widget. Thus our internal function will return a Text() widget with the current quote string placed inside it. But as we said before, the map function returns an object of the Iterable class, and the children property only accepts objects of the ‘List’ class. So we use the ‘toList’ method to convert it into a list.

body: Column(  
 children: **quotes**.map((quote){  
 **return** Text(quote);  
 }).toList(),  
),



Custom Classes:

This is all well and good. But what if we also want to add authors of these quotes? The simplest way to do that is to create a new class for the quotes and then add 2 string variables for the quote and the author. In this manner, we can easily access the quote and its respective author using object.quote and object.author.

So, that’s what we’re gonna do!

We create an entirely new dart file, and call it quote.dart.

Inside it, we create a simple class with 2 string variables for the quote and the author. We also create a constructor for assigning these values. From our java background, we would normally create a class as follows:

**class** Quote  
{  
 String **text**;  
 String **author**;  
  
Quote(String text, String author)  
 {  
 **this**.**text**=text;  
 **this**.**author**=author;  
 }  
  
  
}

Currently, when we create objects, we’ll create them as follows:

Quote myquote = Quote(‘This is the quote’, ‘This is the author’);

While this is perfectly acceptable, we’ll now introduce the concept of a constructor with ‘named’ parameters. Such a constructor allows us to instantiate an object with parameters specified in ANY order. The way to do it is to simply write the constructor as follows:

Quote({**this**.**text**, **this**.**author**});

By doing so, we can instantiate objects like this:

Quote myquote = Quote(text: This is the quote’, author: ‘This is the author’);

AND THIS:

Quote myquote = Quote(author: ‘This is the author’, text: This is the quote’);

Both ways of creating the object are perfectly fine. This is because, it is directly assigning the values to the instance variables as we are explicitly specifying the names of the variables.

Now, do you see something interesting with what we just did? Exactly! This way of creating an object is EXACTLY THE SAME WAY AS WE CREATE A WIDGET AND SPECIFY IT’S PROPERTIES!

So, the properties of widget are nothing but it’s instance variables and we specify them in the widget object along with the names of the parameters JUST BECAUSE THEY HAVE BEEN PREDEFINED WITH A CONSTRUCTOR WITH NAMED PARAMTERS.

Our quote.dart file now looks like:

**class** Quote {  
 String **text**;  
 String **author**;  
  
 Quote({**this**.**text**, **this**.**author**});  
  
}

We now import this class into our main.dart file. Earlier we created a list of strings like:

List<String> **quotes**=[  
 **'If a black cat crosses your path, it signifies that it is going somewhere'**,  
 **'A cat is an example of sophistication minus civilisation'**,  
 **'When a cat closes its eyes, there is a good chance that it cannot see'**];

But now, we replace them with anonymous instances of Quote objects.

List<Quote> **quotes**=[  
 Quote(text:**'If a black cat crosses your path, it signifies that it is going somewhere'**, author: **'Moe the cat'**),  
 Quote(text:**'A cat is an example of sophistication minus civilisation'**, author:**'Anonymous'** ),  
 Quote(text: **'When a cat closes its eyes, there is a good chance that it cannot see'**,author: **'Luis the raccoon'**)  
];

Cool ain’t it? Now, we need to update our map function which was earlier simply outputted the string in the list to a text widget. But, since now the list is no longer of strings, but of Quote objects, we need to explicitly specify that we want to output the text and author.

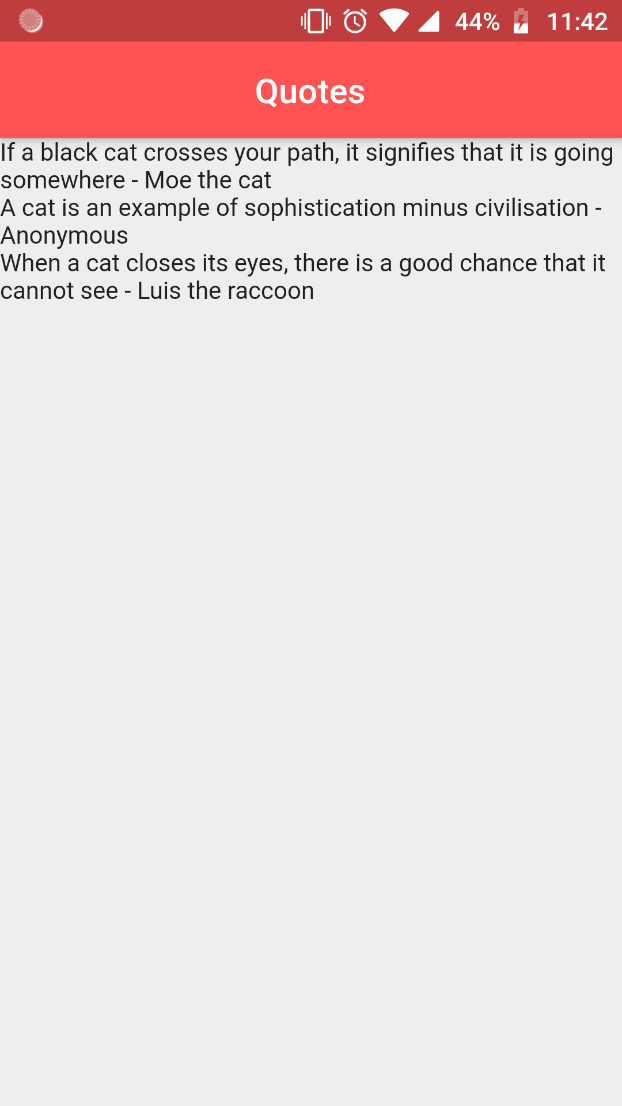
Earlier, the map function looked like:

body: Column(  
 children: **quotes**.map((quote){  
 **return** Text(quote);  
 }).toList(),  
),

We update it to look like:

body: Column(  
 children: **quotes**.map((quote){  
 **return** Text(**'**${quote.**text**} **-** ${quote.**author**}**'**);  
 }).toList(),  
),

One thing that you must have noticed here is that whenever we want to display variables in the middle of a string, we surround them with a dollar ($) sign. But here we want to display object.variable in the middle of the string. For this, we need to surround it with curly braces after the dollar sign.



Cards:

At the moment, this looks ugly. What if we could somehow wrap this entire thing into a ‘Card’, so that it looks cleaner?

Well of course we’ll do that. First we need to create a function which will create a template for the card, and then we will cycle through our list to insert data into the card.

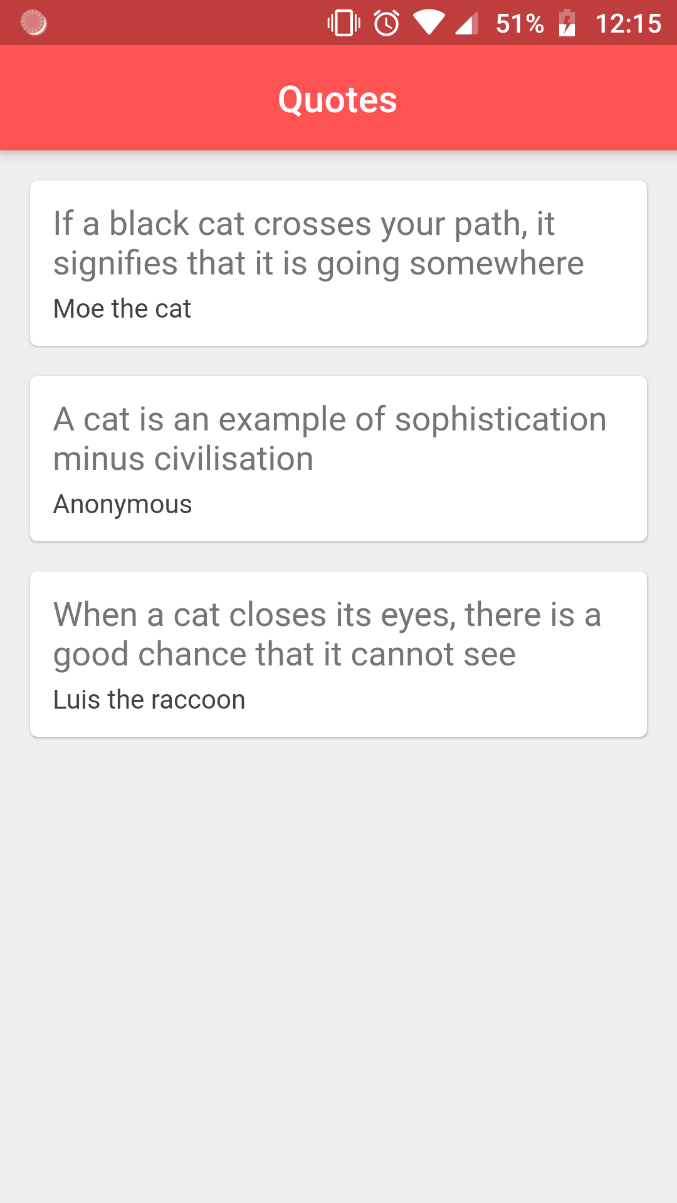
So, we create a function which has a return type of ‘Widget’ and accepts a Quote object as the parameter. This function returns a Card widget, whose properties we will alter. Whatever alterations we do, will be applied to all the cards on the UI.

Our card should contain the quote, and the author i.e. 2 items. So, we’ll need a column widget with 2 text widgets inside it. We also separate these widgets using a SizedBox widget. Our code looks like:

Widget quoteTemplate( Quote quote)  
{  
 **return** Card(  
 margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
 child: Padding(  
 padding: **const** EdgeInsets.all(12.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**stretch**,  
 children: <Widget>[  
 Text(  
 quote.**text**,  
 style: TextStyle(  
 fontSize: 18.0,  
 color: Colors.*grey*[600]  
 ),  
 ),  
 SizedBox(height: 6.0),  
 Text(  
 quote.**author**,  
 style: TextStyle(  
 fontSize: 14.0,  
 color: Colors.*grey*[800]  
 ),  
 )  
 ],  
 ),  
 ),  
 );  
}

We again alter the map function to call the above function for each item in the list. So, essentially the map function accesses one object of the quote object list, it passes that object into the anonymous function. This anonymous function again passes that object into the above created quoteTemplate function. This then returns a ‘Widget’ object in the form of a Card widget. The same is returned by the anonymous function back to the map function which is then put together in a list using the ‘toList’ method.

body: Column(  
 children: **quotes**.map((quote){  
 **return** quoteTemplate(quote);  
 }).toList(),  
),

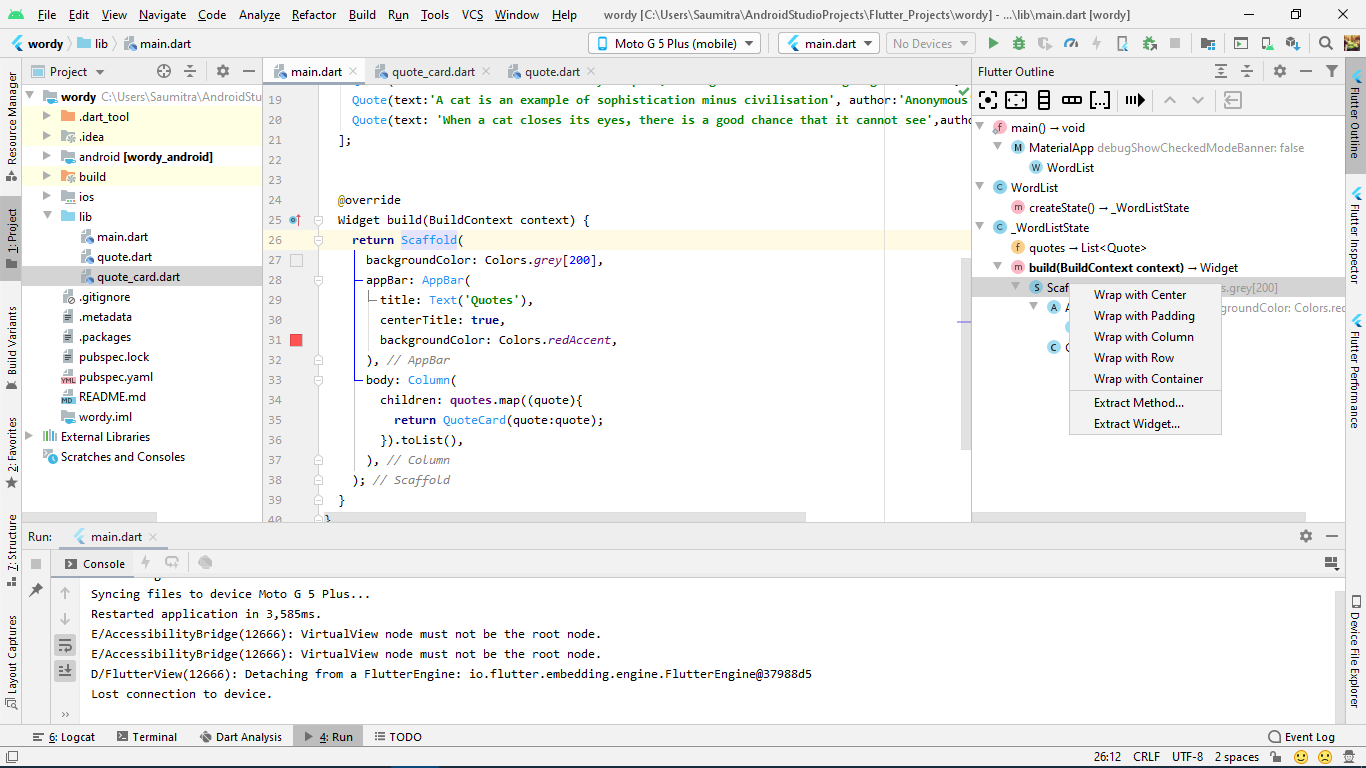


The entire code looks like:

**import 'package:flutter/material.dart'**;  
**import 'quote.dart'**;  
  
**void** main() => runApp(MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
home: WordList(),  
));  
  
**class** WordList **extends** StatefulWidget {  
 @override  
 \_WordListState createState() => \_WordListState();  
}  
  
**class** \_WordListState **extends** State<WordList> {  
  
 List<Quote> **quotes**=[  
 Quote(text:**'If a black cat crosses your path, it signifies that it is going somewhere'**, author: **'Moe the cat'**),  
 Quote(text:**'A cat is an example of sophistication minus civilisation'**, author:**'Anonymous'** ),  
 Quote(text: **'When a cat closes its eyes, there is a good chance that it cannot see'**,author: **'Luis the raccoon'**)  
 ];  
  
 Widget quoteTemplate( Quote quote)  
 {  
 **return** Card(  
 margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
 child: Padding(  
 padding: **const** EdgeInsets.all(12.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**stretch**,  
 children: <Widget>[  
 Text(  
 quote.**text**,  
 style: TextStyle(  
 fontSize: 18.0,  
 color: Colors.*grey*[600]  
 ),  
 ),  
 SizedBox(height: 6.0),  
 Text(  
 quote.**author**,  
 style: TextStyle(  
 fontSize: 14.0,  
 color: Colors.*grey*[800]  
 ),  
 )  
 ],  
 ),  
 ),  
 );  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 backgroundColor: Colors.*grey*[200],  
 appBar: AppBar(  
 title: Text(**'Quotes'**),  
 centerTitle: **true**,  
 backgroundColor: Colors.*redAccent*,  
 ),  
 body: Column(  
 children: **quotes**.map((quote){  
 **return** quoteTemplate(quote);  
 }).toList(),  
 ),  
 );  
 }  
}

Extracting Widgets:

Now that we have beautified our quotes, it is time to make our code more modular. Now, the card widget that we made earlier, we can extract it entirely into a separate stateless widget. Android studio provides a neat way to do this. We simply click on our Card widget in the Flutter Outline



Our earlier code looked like:

Widget quoteTemplate( Quote quote)  
 {  
 **return** Card(  
 margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
 child: Padding(  
 padding: **const** EdgeInsets.all(12.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**stretch**,  
 children: <Widget>[  
 Text(  
 quote.**text**,  
 style: TextStyle(  
 fontSize: 18.0,  
 color: Colors.*grey*[600]  
 ),  
 ),  
 SizedBox(height: 6.0),  
 Text(  
 quote.**author**,  
 style: TextStyle(  
 fontSize: 14.0,  
 color: Colors.*grey*[800]  
 ),  
 )  
 ],  
 ),  
 ),  
 );  
 }

Let’s name our widget as ‘QuoteCard’. Android Studio automatically separates the widget into a completely different stateless widget class with the name ‘QuoteCard’. Our problem here is that because it is now a different class, it doesn’t have access to the quote object that we used to pass into the ‘quoteTemplate’ function. So, we create a local instance of the Quote object, and create a named constructor for it. We then pass ‘quote’ into it, and our problem will be solved.

Our class looks like:

**class** QuoteCard **extends** StatelessWidget {  
  
 **final** Quote **quote**;  
 QuoteCard({**this**.**quote**});  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Card(  
 margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
 child: Padding(  
 padding: **const** EdgeInsets.all(12.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.**stretch**,  
 children: <Widget>[  
 Text(  
 **quote**.**text**,  
 style: TextStyle(  
 fontSize: 18.0,  
 color: Colors.*grey*[600]  
 ),  
 ),  
 SizedBox(height: 6.0),  
 Text(  
 **quote**.**author**,  
 style: TextStyle(  
 fontSize: 14.0,  
 color: Colors.*grey*[800]  
 ),  
 )  
 ],  
 ),  
 ),  
 );  
 }  
}

Our quoteTemplate function now looks like:

Widget quoteTemplate( Quote quote)  
 {  
 **return** QuoteCard(quote)

}

The quoteTemplate function now however looks worthless, as we are redundantly passing quote object into it as we have already created our own widget which accepts a quote object. So, we delete it, and make changes directly to the map function.

In the map function, instead of calling the quoteTemplate function, we can now directly create the card widget as we have extracted it. This is similar to the earlier way where there was only the ‘Text’ widget.

The map function now looks like:

body: Column(  
 children: **quotes**.map((quote){  
 **return** QuoteCard(quote:quote);  
 }).toList(),  
),

We could do one last thing now. In order to make our widget truly reusable, we could create our own new dart file which has only the widget. We can import this dart file into our code, AS WELL AS any other code that we may wish to add to this project. So, currently we have 3 dart files.

1. Main.dart:

**import 'package:flutter/material.dart'**;  
**import 'quote.dart'**;  
**import 'quote\_card.dart'**;  
  
**void** main() => runApp(MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
home: WordList(),  
));  
  
**class** WordList **extends** StatefulWidget {  
 @override  
 \_WordListState createState() => \_WordListState();  
}  
  
**class** \_WordListState **extends** State<WordList> {  
  
 List<Quote> **quotes**=[  
 Quote(text:**'If a black cat crosses your path, it signifies that it is going somewhere'**, author: **'Moe the cat'**),  
 Quote(text:**'A cat is an example of sophistication minus civilisation'**, author:**'Anonymous'** ),  
 Quote(text: **'When a cat closes its eyes, there is a good chance that it cannot see'**,author: **'Luis the raccoon'**)  
 ];  
  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 backgroundColor: Colors.*grey*[200],  
 appBar: AppBar(  
 title: Text(**'Quotes'**),  
 centerTitle: **true**,  
 backgroundColor: Colors.*redAccent*,  
 ),  
 body: Column(  
 children: **quotes**.map((quote){  
 **return** QuoteCard(quote:quote);  
 }).toList(),  
 ),  
 );  
 }  
}

1. Quote.dart

**class** Quote {  
 String **text**;  
 String **author**;  
  
 Quote({**this**.**text**, **this**.**author**});  
  
}

1. Quote\_card.dart
2. **import 'package:flutter/material.dart'**;  
   **import 'quote.dart'**;  
     
   **class** QuoteCard **extends** StatelessWidget {  
     
    **final** Quote **quote**;  
    QuoteCard({**this**.**quote**});  
     
    @override  
    Widget build(BuildContext context) {  
    **return** Card(  
    margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
    child: Padding(  
    padding: **const** EdgeInsets.all(12.0),  
    child: Column(  
    crossAxisAlignment: CrossAxisAlignment.**stretch**,  
    children: <Widget>[  
    Text(  
    **quote**.**text**,  
    style: TextStyle(  
    fontSize: 18.0,  
    color: Colors.*grey*[600]  
    ),  
    ),  
    SizedBox(height: 6.0),  
    Text(  
    **quote**.**author**,  
    style: TextStyle(  
    fontSize: 14.0,  
    color: Colors.*grey*[800]  
    ),  
    )  
    ],  
    ),  
    ),  
    );  
    }  
   }

Functions as Parameters:

Let’s add a final finishing touch to our app. Let’s add a delete button to the card which on clicking will delete the quote from the list.

Here however we have a problem. Because we have created 2 different dart files for the card widget and the home page scaffold, and the list of objects is in the home dart file, the card widget does not have access to the list. But, the onPressed property is in the card widget dart file.

In this case then, we create a function to delete the quote IN THE MAIN file, and then pass the function as a parameter to the card widget so that the function can be simply called when the button is pressed.

Our widget now has 2 properties which we are passing from the main.dart file:

1. The quote object (as a reference in order to access the quote and author)
2. The delete function which we’ll define here itself (so that it can simply be called in the onPressed property of the button)

The delete function is pretty straightforward. We simply change the state of the app to remove an object from the quotes list. We’ll use the setState function in order to re-render the screen after the quote has been removed. Inside it, we’ll simply delete the current cycled element from the list by calling the ‘quotes.remove(quote)’.

We shall be passing all this in map function. Let’s see how the map function looks like:

body: Column(  
 children: **quotes**.map((quote){  
 **return** QuoteCard(  
 quote:quote,  
 delete:(){  
 setState(() {  
 **quotes**.remove(quote);  
 });  
 });  
 }).toList(),  
),

Now, since we have passed a function as a parameter, we’ll need to update our constructor accordingly. An interesting thing about Dart is that EVEN FUNCTIONS ARE OBJECTS. Thus the datatype of functions is ‘Function’ (Cool ain’t it?)

So, our Card Class looks like:

**class** QuoteCard **extends** StatelessWidget {  
  
 **final** Quote **quote**;  
 **final** Function **delete**;  
 QuoteCard({**this**.**quote**,**this**.**delete**});

…

}

Our 3 dart files now are as follows:

1. Main.dart

**import 'package:flutter/material.dart'**;  
**import 'quote.dart'**;  
**import 'quote\_card.dart'**;  
  
**void** main() => runApp(MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
home: WordList(),  
));  
  
**class** WordList **extends** StatefulWidget {  
 @override  
 \_WordListState createState() => \_WordListState();  
}  
  
**class** \_WordListState **extends** State<WordList> {  
  
 List<Quote> **quotes**=[  
 Quote(text:**'If a black cat crosses your path, it signifies that it is going somewhere'**, author: **'Moe the cat'**),  
 Quote(text:**'A cat is an example of sophistication minus civilisation'**, author:**'Anonymous'** ),  
 Quote(text: **'When a cat closes its eyes, there is a good chance that it cannot see'**,author: **'Luis the raccoon'**)  
 ];  
  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 backgroundColor: Colors.*grey*[200],  
 appBar: AppBar(  
 title: Text(**'Quotes'**),  
 centerTitle: **true**,  
 backgroundColor: Colors.*redAccent*,  
 ),  
 body: Column(  
 children: **quotes**.map((quote){  
 **return** QuoteCard(  
 quote:quote,  
 delete:(){  
 setState(() {  
 **quotes**.remove(quote);  
 });  
 });  
 }).toList(),  
 ),  
 );  
 }  
}

1. Quote.dart

**class** Quote {  
 String **text**;  
 String **author**;  
  
 Quote({**this**.**text**, **this**.**author**});  
  
}

1. Quote\_card.dart
2. **import 'package:flutter/material.dart'**;  
   **import 'quote.dart'**;  
     
   **class** QuoteCard **extends** StatelessWidget {  
     
    **final** Quote **quote**;  
    **final** Function **delete**;  
    QuoteCard({**this**.**quote**,**this**.**delete**});  
     
    @override  
    Widget build(BuildContext context) {  
    **return** Card(  
    margin: EdgeInsets.fromLTRB(16.0, 16.0, 16.0, 0.0),  
    child: Padding(  
    padding: **const** EdgeInsets.all(12.0),  
    child: Column(  
    crossAxisAlignment: CrossAxisAlignment.**stretch**,  
    children: <Widget>[  
    Text(  
    **quote**.**text**,  
    style: TextStyle(  
    fontSize: 18.0,  
    color: Colors.*grey*[600]  
    ),  
    ),  
    SizedBox(height: 6.0),  
    Text(  
    **quote**.**author**,  
    style: TextStyle(  
    fontSize: 14.0,  
    color: Colors.*grey*[800]  
    ),  
    ),  
    SizedBox(height: 8.0),  
    FlatButton.icon(  
    onPressed: **delete**,  
    icon: Icon(  
    Icons.*delete*,  
    color: Colors.*red*,  
    ),  
    label: Text(  
    **'Delete Quote'**,  
    style: TextStyle(  
    color: Colors.*red* ),  
    )  
    )  
    ],  
    ),  
    ),  
    );  
    }  
   }

The output looks like:

