

Chapter 8: User Inputs and Form

8.1. Showing snackbar and Alert dialog

Snackbars are often useful to convey information about a specific action to the users. For example, If a user added an item to cart in an ecommerce application a snackbar saying, “Item added to cart” can be used.

Following steps can be followed to create a Snackbar:

- Create a Scaffold.
- Display Snackbar.
- Provide optional action.

Create a Scaffold

To create a Snackbar in flutter you will have to first create a Scaffold and then add a Snackbar to it.

```
Scaffold(  
  appBar: AppBar(  
    title: Text('SnackBar Demo'),  
  ),  
  body: APageThatWillHaveSnackBar(),  
);
```

Display Snackbar

With Scaffold ready, a Snackbar can be displayed using a ScaffoldMessenger.

```
final snackBar = SnackBar(content: Text('Yay! A SnackBar!'));  
  
// Find the ScaffoldMessenger in the widget tree  
// and use it to show a SnackBar.  
ScaffoldMessenger.of(context).showSnackBar(snackBar);
```

Provide an action (Optional)

Adding an action inside a snackbar is optional but often can come in handy.

```
final snackBar = SnackBar(  
  content: Text('Yay! A SnackBar!'),  
  action: SnackBarAction(  
    label: 'Undo',  
    onPressed: () {  
      // Some code to undo the change.  
    },  
  ),  
);
```

Alert dialog

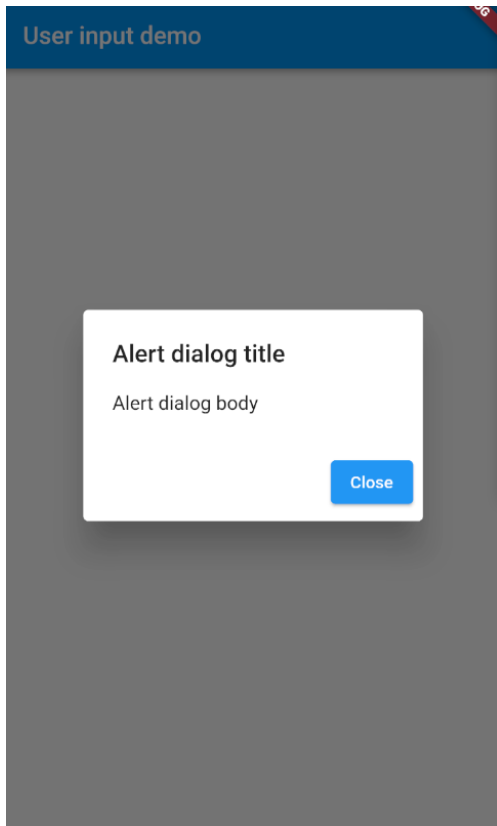
An alert dialog is a useful tool that alerts the app's user. It is a pop up in the middle of the screen which places an overlay over the background. Most commonly, it is used to confirm one of the user's potentially irreversible actions.

Example:



```
1 class MainScreen extends StatelessWidget {
2   const MainScreen({
3     super.key,
4   });
5
6   @override
7   Widget build(BuildContext context) {
8     return Scaffold(
9       appBar: AppBar(
10        title: const Text("User input demo"),
11      ),
12      body: Center(
13        child: ElevatedButton(
14          onPressed: () {
15            showDialog(
16              context: context,
17              builder: (_) {
18                return AlertDialog(
19                  title: const Text("Alert dialog title"),
20                  content: const Text("Alert dialog body"),
21                  actions: [
22                    ElevatedButton(
23                      onPressed: () {
24                        Navigator.of(context).pop();
25                      },
26                      child: const Text('Close'))
27                  ],
28                );
29              });
30        },
31        child: const Text("Click me")),
32      ),
33    );
34  }
35 }
36
```

Result:



8.2. Flutter Form

Forms are an integral part of all modern mobile and web applications. It is mainly used to interact with the app as well as gather information from the users. They can perform many tasks, which depend on the nature of your business requirements and logic, such as authentication of the user, adding user, searching, filtering, ordering, booking, etc. A form can contain text fields, buttons, checkboxes, radio buttons, etc.

Creating Form

Flutter provides a **Form widget** to create a form. The form widget acts as a container, which allows us to group and validate the multiple form fields. When you create a form, it is necessary to provide the **GlobalKey**. This key uniquely identifies the form and allows you to do any validation in the form fields.

The form widget uses child widget **TextFormField** to provide the users to enter the text field. This widget renders a material design text field and also allows us to display validation errors when they occur.

Let us create a form. First, create a Flutter project and replace the following code in the main.dart file. In this code snippet, we have created a custom class named **MyCustomForm**. Inside this class, we define a global key as **_formKey**. This key holds a **FormState** and can be used to retrieve the form widget. Inside the **build** method of this class, we have added some custom style and use the TextFormField widget to provide the form fields such as name, phone number, date of birth, or just a normal field. Inside the TextFormField, we have used **InputDecoration** that provides the look and feel of your form properties such as borders, labels, icons, hint, styles, etc. Finally, we have added a **button** to submit the form.

```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    const appTitle = 'Flutter Form Demo';
    return MaterialApp(
      title: appTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(appTitle),
        ),
        body: const MyCustomForm(),
      ),
    );
  }
}

// Create a Form widget.
class MyCustomForm extends StatefulWidget {
  const MyCustomForm({super.key});

  @override
  MyCustomFormState createState() {
    return MyCustomFormState();
  }
}
```

```
}  
}
```

// Create a corresponding State class. This class holds data related to the form.

```
class MyCustomFormState extends State<MyCustomForm> {  
  // Create a global key that uniquely identifies the Form widget  
  // and allows validation of the form.  
  final _formKey = GlobalKey<FormState>();
```

```
@override
```

```
Widget build(BuildContext context) {  
  // Build a Form widget using the _formKey created above.  
  return Padding(  
    padding: const EdgeInsets.all(16.0),  
    child: Form(  
      key: _formKey,  
      child: Column(  
        crossAxisAlignment: CrossAxisAlignment.start,  
        children: <Widget>[  
          TextFormField(  
            decoration: const InputDecoration(  
              icon: Icon(Icons.person),  
              hintText: 'Enter your name',  
              labelText: 'Name',  
            ),  
          ),  
          TextFormField(  
            decoration: const InputDecoration(  
              icon: Icon(Icons.phone),  
              hintText: 'Enter a phone number',  
              labelText: 'Phone',  
            ),  
          ),  
          TextFormField(  
            decoration: const InputDecoration(  
              icon: Icon(Icons.calendar_today),  
              hintText: 'Enter your date of birth',  
              labelText: 'Dob',  
            ),  
          ),  
        ],  
      ),  
    ),  
  );  
}
```

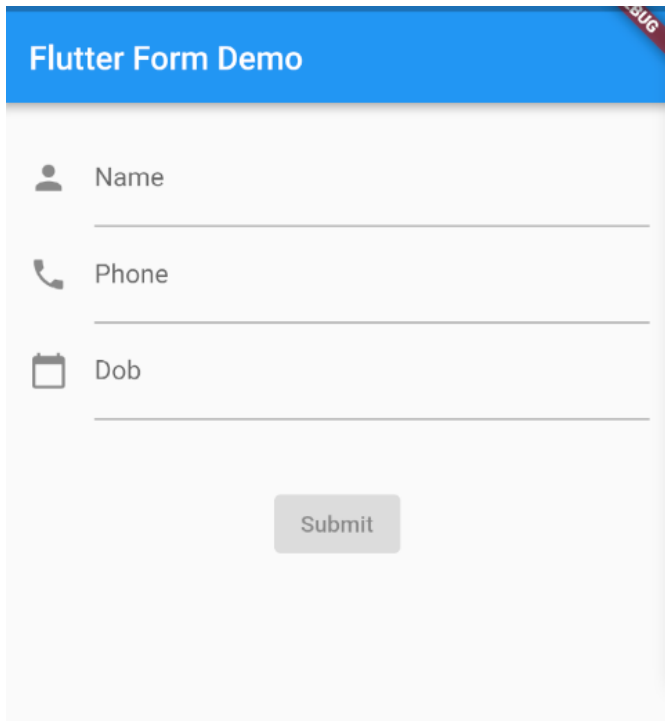
```

    ),
    Container(
      padding: const EdgeInsets.only(left: 150.0, top: 40.0),
      child: const ElevatedButton(
        onPressed: null,
        child: Text('Submit'),
      )),
  ],
),
),
);
}
}

```

Output

Now, run the app, you can see the following screen in your Android Emulator. This form contains three field name, phone number, date of birth, and submit button.



The screenshot shows a mobile application interface titled "Flutter Form Demo". It features a light gray background with three form fields, each preceded by a small icon: a person icon for "Name", a telephone icon for "Phone", and a calendar icon for "Dob". Each field has a horizontal input line. At the bottom center, there is a gray "Submit" button. A red "BUG" sticker is visible in the top right corner of the app's header.

Form validation

Validation is a method, which allows us to correct or confirms a certain standard. It ensures the authentication of the entered data. Validating forms is a common practice in all digital interactions. To validate a form in a flutter, we need to implement mainly three steps.

Step 1: Use the Form widget with a global key.

Step 2: Use TextFormField to give the input field with validator property.

Step 3: Create a button to validate form fields and display validation errors.

Let us understand it with the following example. In the above code, we have to use **validator()** function in the TextFormField to validate the input properties. If the user gives the wrong input, the validator function returns a string that contains an **error message**; otherwise, the validator function return **null**. In the validator function, make sure that the TextFormField is not empty. Otherwise, it returns an error message.

The validator() function can be written as below code snippets:

```
validator: (value) {  
  if (value.isEmpty) {  
    return 'Please enter some text';  
  }  
  return null;  
},
```

Now, open the **main.dart** file and add validator() function in the TextFormField widget. Replace the following code with the main.dart file.

```
import 'package:flutter/material.dart';  
  
void main() => runApp(const MyApp());  
  
class MyApp extends StatelessWidget {
```



```
const MyApp({super.key});

@override

Widget build(BuildContext context) {

  const appTitle = 'Flutter Form Demo';

  return MaterialApp(

    title: appTitle,

    home: Scaffold(

      appBar: AppBar(

        title: const Text(appTitle),

      ),

      body: const MyCustomForm(),

    ),

  );

}

// Create a Form widget.

class MyCustomForm extends StatefulWidget {

  const MyCustomForm({super.key});

  @override

  MyCustomFormState createState() {
```

```

        return MyCustomFormState();
    }
}

// Create a corresponding State class, which holds data related to the
// form.

class MyCustomFormState extends State<MyCustomForm> {

    // Create a global key that uniquely identifies the Form widget

    // and allows validation of the form.

    final _formKey = GlobalKey<FormState>();

    @override

    Widget build(BuildContext context) {

        // Build a Form widget using the _formKey created above.

        return Form(

            key: _formKey,

            child: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: <Widget>[

                    TextFormField(

                        decoration: const InputDecoration(

                            icon: Icon(Icons.person),

                            hintText: 'Enter your full name',

                            labelText: 'Name',

```

```
    ),  
  
    validator: (value) {  
  
        if (value!.isEmpty) {  
  
            return 'Please enter some text';  
  
        }  
  
        return null;  
  
    },  
  
),  
  
TextFormField(  
  
    decoration: const InputDecoration(  
  
        icon: Icon(Icons.phone),  
  
        hintText: 'Enter a phone number',  
  
        labelText: 'Phone',  
  
    ),  
  
    validator: (value) {  
  
        if (value!.isEmpty) {  
  
            return 'Please enter valid phone number';  
  
        }  
  
        return null;  
  
    },  
  
),  
  
TextFormField(  
  
    decoration: const InputDecoration(  
  

```

```

        icon: Icon(Icons.calendar_today),

        hintText: 'Enter your date of birth',

        labelText: 'Dob',

    ),

    validator: (value) {

        if (value!.isEmpty) {

            return 'Please enter valid date';

        }

        return null;

    },

),

Container(

    padding: const EdgeInsets.only(left: 150.0, top: 40.0),

    child: ElevatedButton(

        child: const Text('Submit'),

        onPressed: () {

            // It returns true if the form is valid,

            // otherwise returns false

            if (_formKey.currentState!.validate()) {

                // If the form is valid, display a Snackbar.

                ScaffoldMessenger.of(context).showSnackBar(const

SnackBar(

                    content: Text('Data is in processing.')));

            }

```

```

        },
      )),
    ],
  ),
);
}
}

```

Output

Now, run the app. The following screen appears.



In this form, if you left any input field blank, you will get an error message like below screen.

