23. Builders

Introduction

The purpose of this chapter is to learn how to use Flutter builder classes.

If you go to the official Flutter documentation for the builder class you see it has the following description:

“A platonic widget that calls a closure to obtain its child widget.”

What does that mean? ☺

What is a Builder?

The term closure is just another name for a lambda function, an anonymous method.
So, builder is really a lambda that acts similarly to the Widget’s build method:

- You pass it a BuildContext and any other variables you need to.
- It returns a Widget.

How Do You Use a Builder?

Instead of passing a Widget back from your build method, instead you pass back an anonymous builder function that takes whatever parameters are required (including the BuildContext) and spits out a Widget.

Nested Builders

You can nest builders inside builders and this (although sometimes complicated) works very well. There is an example in this Chapter called ‘Multiple Builders’, which uses nested builders.

Common Builders

AnimatedBuilder

We will cover this builder in the Animations chapter.

GridView Builder

Similar to the ListView builder. Quite often you will end up with large dynamic data grids and you need to display them onscreen using a Grid, even though the user may not scroll all the way to the bottom.

If you simply add a Widget for each item in the grid, you end up with a huge amount of child Widgets, most of which will never be seen. This is not efficient.

This is where the GridView builder comes in. When the user scrolls down through the grid, the GridView builder is invoked to create the child widgets when they are needed, not ahead of time. Much more efficient.

You write a GridView builder and specify it to the GridView in the ‘itemBuilder’ argument in the constructor. In the builder method, you accept BuildContext and index arguments and you spit out a Widget. This is perfect if your data is held in array – all you do is get the data for that item from the array using that index.

There is an example in this Chapter called ‘Multiple Builders’. It uses the GridView builder, amongst other builders!

FutureBuilder

FutureBuilder is a widget that returns another widget based on the Future’s execution result. It serves as a bridge between Futures and the Widget’s UI.

Example – ‘future_builder_app’

This app uses a FutureBuilder to calculates a bunch of timestamps using a Future computation and display it. The screen is blank for a few seconds then it displays a list of times. It’s not terribly pretty!


6:57
Future
hit the button to show calculation

Source Code:

```
import 'dart:async';

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
```

```
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Future Builder App',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new HomeWidget(),
    );
  }
}

class HomeWidget extends StatefulWidget {
  String computeListOfTimestamps(int count) {
    StringBuffer sb = new StringBuffer();
    for (int i = 0; i < count; i++) {
      sb.writeln("${i + 1} : ${DateTime.now()}");
    }
    return sb.toString();
  }

  Future<String> createFutureCalculation(int count) {
    return new Future(() {
      return computeListOfTimestamps(count);
    });
  }
```

```
  HomeWidget({Key key}) : super(key: key);

  @override
  _HomeWidgetState createState() => new
_HomeWidgetState();
}

class _HomeWidgetState extends State<HomeWidget> {
  bool _showCalculation = false;

  void _onInvokeFuturePressed() {
    setState(() {
      _showCalculation = !_showCalculation;
    });
  }

  @override
  Widget build(BuildContext context) {
    Widget child = _showCalculation
        ? FutureBuilder(
            future: widget.createFutureCalculation(10000),
            builder: (BuildContext context, AsyncSnapshot
snapshot) {
              return Expanded(
                  child: SingleChildScrollView(
                      child: Text(
                          '${snapshot.data == null ? "" :
snapshot.data}',
```

```
                      style: TextStyle(fontSize: 20.0))));
           })
        : Text('hit the button to show calculation');
    return new Scaffold(
      appBar: new AppBar(
        title: new Text("Future"),
      ),
      body: new Center(
          child: new Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: <Widget>[child])),
      floatingActionButton: new FloatingActionButton(
        onPressed: _onInvokeFuturePressed,
        tooltip: 'Invoke Future',
        child: new Icon(Icons.refresh),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

ListView Builder:

Similar to the GridView builder. Quite often you will end up with large dynamic lists of data and you need to display them onscreen using a ListView, even though the user may not scroll through the list.

If you simply add a Widget for each item in the list, you end up with a huge amount of child Widgets, most of which will never be seen. This is not efficient. This is where the ListView builder comes in. When the user scrolls through the list, the ListView builder is invoked to create the child widgets when they are needed, not ahead of time. Much more efficient.

You write a ListView builder and specify it to the ListView in the 'itemBuilder' argument in the constructor. In the builder method, you accept BuildContext and index arguments and you spit out a Widget. This is perfect if your data is held in array – all you do is get the data for that item from the array using that index.

Example – 'listview_builder'

This app shoes a list of NASA offices in the US. The app sorts the list of NASA offices by name in the constructor and prints to the console everytime the ListView builder is invoked, so you can see how the child widgets are built 'on demand'. It also displays each Nasa Office in a ListTile.

Source Code

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
```

```
    theme: new ThemeData(
      primarySwatch: Colors.blue,
    ),
    home: new MyHomeWidget(),
   );
  }
}

class MyHomeWidget extends StatelessWidget {
  List<dynamic> _nasaOffices = [
   {
    "Name": "Mach 6, High Reynolds Number Facility",
    "Address": "1864 4th St",
    "City": "Wright-Patterson AFB",
    "State": "OH",
    "ZIP": "45433-7541",
    "Country": "US"
   },
```

... edited for bevity …

```
{
    "Name": "N206A - 12 FOOT PRESSURE WIND TUNNEL
AUXILIARIES (PAPAC)",
    "Address": "Code RC",
    "City": "Moffett Field",
    "State": "CA",
    "ZIP": "94035",
```

```
    "Country": "US"
   }
  ];

  MyHomeWidget({Key key}) : super(key: key) {
    _nasaOffices.sort((a, b) =>
a['Name'].compareTo(b['Name']));
  }

  @override
  Widget build(BuildContext context) {
    ListView builder = ListView.builder(
        itemCount: _nasaOffices.length,
        itemBuilder: (context, index) {
          print('invoking itemBuilder for row ${index}');
          var nasaOffice = _nasaOffices[index];
          return ListTile(
              title: Text('${nasaOffice['Name']}'),
              subtitle: Text('${nasaOffice['Address']},
${nasaOffice['City']},'
                  '${nasaOffice['State']}, ${nasaOffice['ZIP']},'
                  '${nasaOffice['Country']}'),
              trailing: Icon(Icons.arrow_right));
        });
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Nasa Offices"),
```

```
        ),
        body: new Center(child: builder));
  }
}
```

OrientationBuilder

Sometime the user will rotate their screen from portrait mode to landscape mode and visa-versa. You may wish to change the layout to take advantage of the extra space. For example, you may want to show a grid with 2 items across in portrait, 3 items across in landscape.

This is where the OrientationBuilder comes in. Wrap your builder code in an OrientationBuilder and it can react to orientation changes.

There is an example in this Chapter called ‘Multiple Builders’. It uses the OrientationBuilder, amongst other builders!

PageRoutebuilder

We will cover this builder in the Routing & Navigation chapter.

StreamBuilder

StreamBuilder

StreamBuilders listen for changes in streams and build Widgets when the stream data changes. Thus, your Widgets can update when the state changes and the state change is pushed to a stream.

Some of the state management patterns (such as the BLoC, covered later on in its own chapter) use this builder to update the ui when a stream value changes.

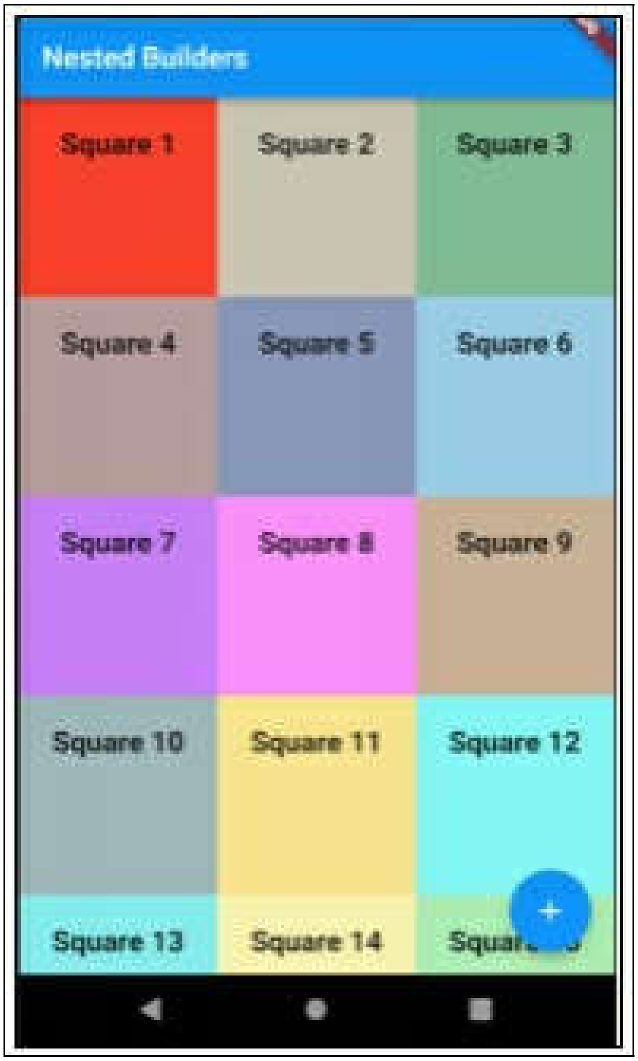
There is an example in this Chapter called ‘Multiple Builders’. It uses the StreamBuilder, amongst other builders!

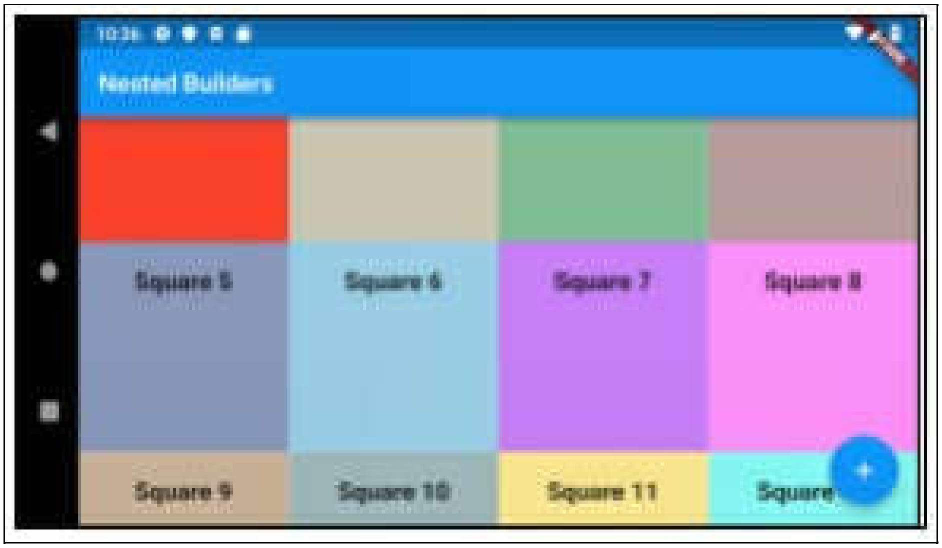
Example – ‘nested_builders’

This app shows some colored squares: 3 across in portrait, 4 across in landscape. It also allows you to hit the ‘+’ button to add more squares. To do this, the app stores its state (the squares) in a BLoC (don’t worry about this too much, we will cover this in another chapter) and uses the following builders in the HomeWidget, nested within each other:

- StreamBuilder – update ui when state changes

- OrientationBuilder – update ui when orientation changes
  - GridView Builder – builds ui for grid.





Source Code

```
import 'dart:async';
import 'dart:math';

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

class Square {
  String _text;
  Color _color;

  Square(this._text, this._color);

  operator ==(other) =>
      (other is Square) && (_text == other._text) && (_color ==
other._color);

  int get hashCode => _text.hashCode ^ _color.hashCode;
```

```
  Color get color => _color;
  String get text => _text;
}

class Bloc {
  // BLoC stands for Business Logic Component.
  final _random = new Random();
  List<Square> _squareList = [];

  Bloc() {
    _addActionStreamController.stream.listen(_handleAdd);
  }

  int next(int min, int max) => min + _random.nextInt(max -
min);

  List<Square> initSquareList() {
    _squareList = [new Square("Square 1", Colors.red)];
    return _squareList;
  }

  void dispose() {
    _addActionStreamController.close();
  }

  Square createSquare() {
    String nextSquareNumberAsString = (_squareList.length +
1).toString();
    return Square("Square " +
```

```
nextSquareNumberAsString.toString(),
        Color.fromRGBO(next(0, 255), next(0, 255), next(0,
255), 0.5));
  }

  void _handleAdd(void v) {
    _squareList.add(createSquare());
    _squareListSubject.add(_squareList);
  }

  // Streams for State Updates
  Stream<List<Square>> get squareListStream =>
_squareListSubject.stream;
  final _squareListSubject = BehaviorSubject<List<Square>>
();

  // Sinks for Actions
  Sink get addAction => _addActionStreamController.sink;
  final _addActionStreamController = StreamController();
}

class BlocProvider extends InheritedWidget {
  final Bloc bloc;

  BlocProvider({
    Key key,
    @required this.bloc,
    Widget child,
  }) : super(key: key, child: child);
```

```
  @override
  bool updateShouldNotify(InheritedWidget oldWidget) =>
true;

  static Bloc of(BuildContext context) =>
      (context.inheritFromWidgetOfExactType(BlocProvider) as
BlocProvider).bloc;
}

void main() => runApp(new NestedBuildersAppWidget());

class NestedBuildersAppWidget extends StatelessWidget {
  final Bloc _bloc = new Bloc();

  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Nested Builders',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: BlocProvider(
        bloc: _bloc,
        child: new HomeWidget(title: 'Nested Builders'),
      ),
    );
  }
}
```

```
class HomeWidget extends StatelessWidget {
  HomeWidget({Key key, this.title}) : super(key: key);

  final String title;

  @override
  Widget build(BuildContext context) {
    final bloc = BlocProvider.of(context);
    return new Scaffold(
      appBar: new AppBar(
        title: new Text(title),
        actions: <Widget>[],
      ),
      body: StreamBuilder<List<Square>>(
          stream: bloc.squareListStream,
          initialData: bloc.initSquareList(),
          builder: (context, snapshot) {
            List<Square> squares = snapshot.data;
            return OrientationBuilder(builder: (context, 
orientation) {
              return GridView.builder(
                  itemCount: squares.length,
                  gridDelegate: new 
SliverGridDelegateWithFixedCrossAxisCount(
                      crossAxisCount:
                          (orientation == Orientation.portrait) ? 3 : 
4),
                  itemBuilder: (BuildContext context, int index) {
```

```
                    return new GridTile(
                        child: Container(
                            color: squares[index].color,
                            child: Padding(
                                padding: EdgeInsets.all(20.0),
                                child: Text(squares[index]._text,
                                    style: TextStyle(
                                        fontSize: 20.0,
                                        fontWeight: FontWeight.bold),
                                    textAlign: TextAlign.center))));
                  });
            });
          }),
      floatingActionButton: new FloatingActionButton(
        onPressed: () => bloc.addAction.add(null),
        tooltip: 'Add',
        child: new Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
  }
}
```

24. Routing & Navigation

Introduction

Navigation is a key part of any mobile app as users will constantly be navigating between different screens, for example, from a customer list to a customer detail screen.

The purpose of this chapter is to learn how to write Flutter apps that include navigation.

Navigator Class

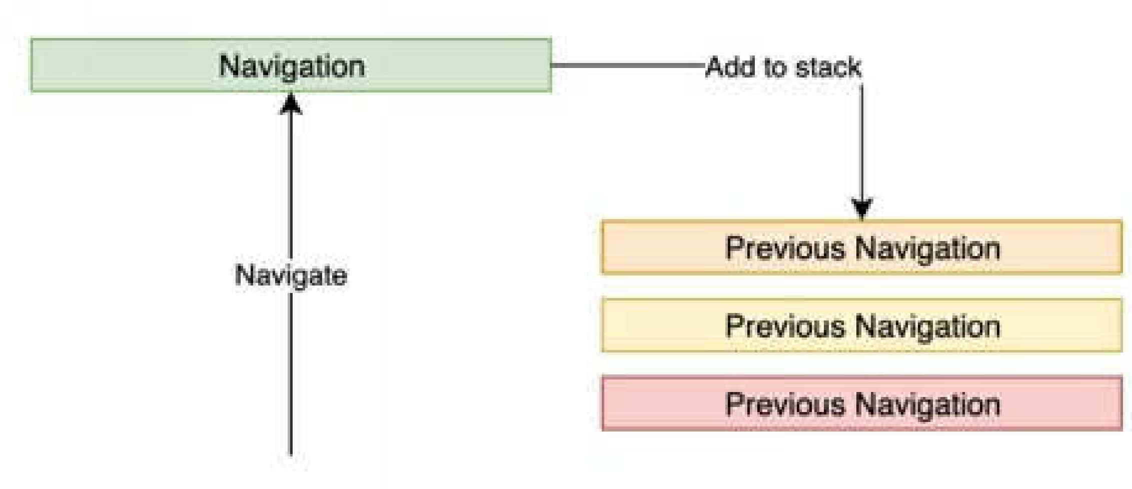
Flutter provides a Navigator class to help us perform navigation in our app.
We can provide Navigation between Widgets with or without named routes.

Stack of Routes

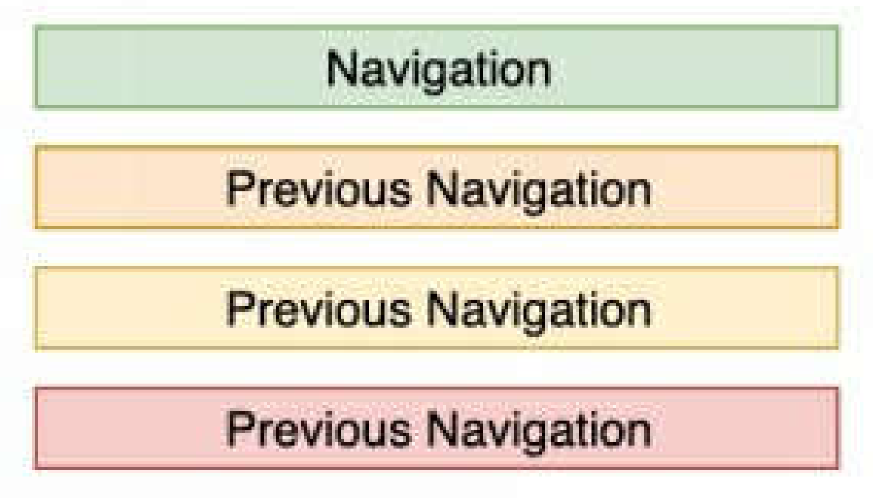
When you start using the Navigator class, you realize that it manages a stack of Routes, a history of visited screens/pages. When you navigate back, you pop a Route off the stack.

Navigating Forward

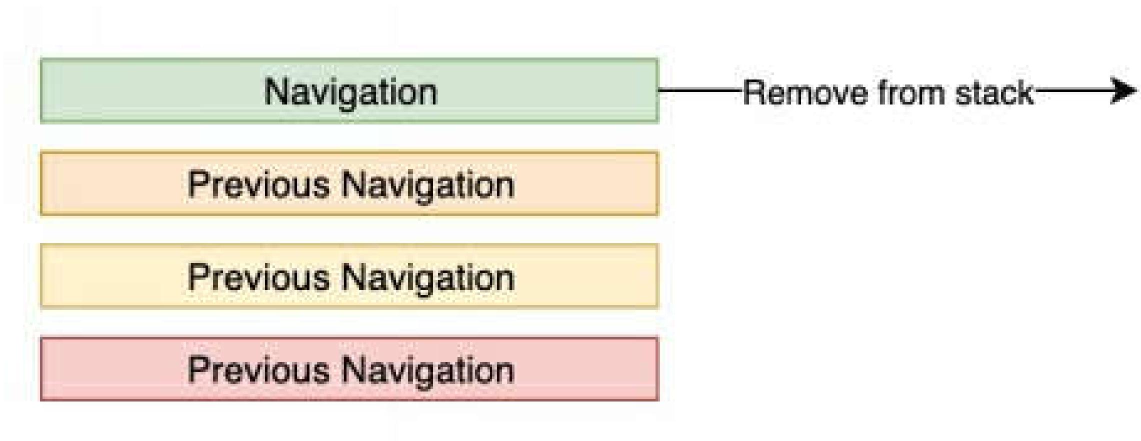
When you navigate forward (for example to a new part of the app), you push a Route to the stack.



Results In:



Navigating Back:



Results In:

Invoking Navigation without Named Routes

This is simple and is a great option, especially for smaller apps without too many Widgets. However, this can result in code duplication if we use this method to navigate to the same Widget in more than one place.

Navigating Forward

Note how we navigate forward in the example:

```
Navigator.push(
      context,
      MaterialPageRoute(builder: (context) =>
CustomerWidget(customer)),
    );
```

We create a new MaterialPageRoute object with a builder that will create the new target Widget to navigate to.

This is another way to do the same thing with a PageRouteBuilder instead creating a MaterialPageRoute:

```
PageRouteBuilder pageRouteBuilder =
PageRouteBuilder(pageBuilder:
    (BuildContext context, Animation animation,
        Animation secondaryAnimation) {
  return CustomerWidget(customer);
});
Navigator.push(
  context,
  pageRouteBuilder,
);
```

Animation

When navigating, MaterialPageRoutes automatically perform animations for us. Different animations that follow the design language of the target platform.

PageRouteBuilder gives us more control over the animations.

Dialog

Note that the MaterialPageRoute also has a ‘fullScreenDialog’ constructor argument. This makes the new target Widget appear as a dialog rather than another Widget. As such it displays a ‘Close’ button instead of a back arrow button.

Navigating Backwards

Note how we don’t need to do anything for the back arrow button to appear on the toolbar. Very nice! That back button simply does a Navigator.pop to navigate the user backwards to the previous navigation.

Data

Passing Data to Target Navigation

We pass the Customer and Order data between widgets using constructors that accept the Customer or Order data. We then push that object to the Navigator stack to navigate forward.

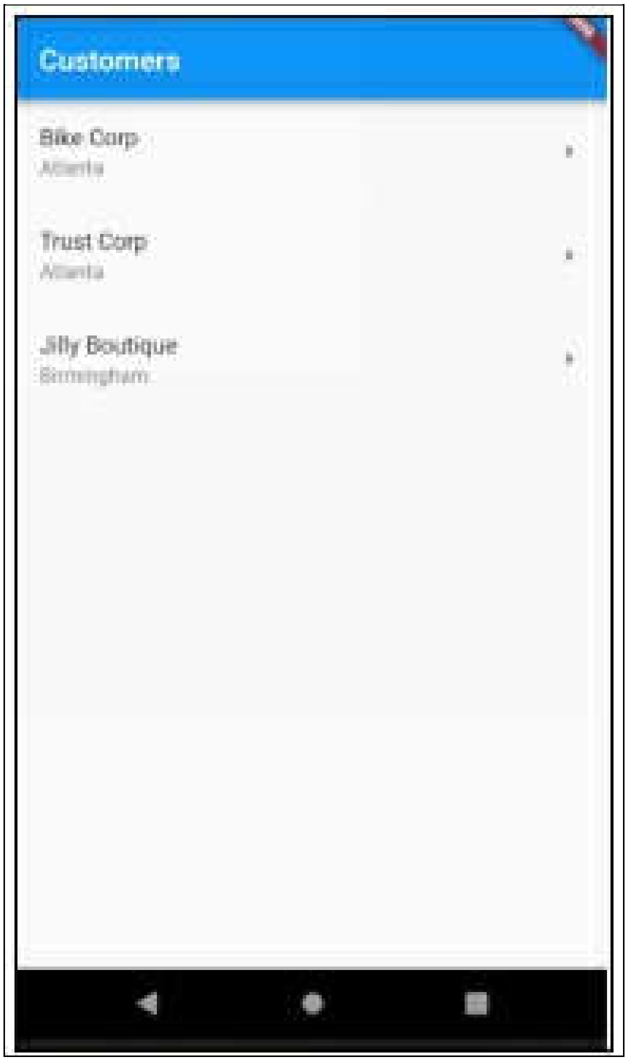
Returning Data from Target Navigation

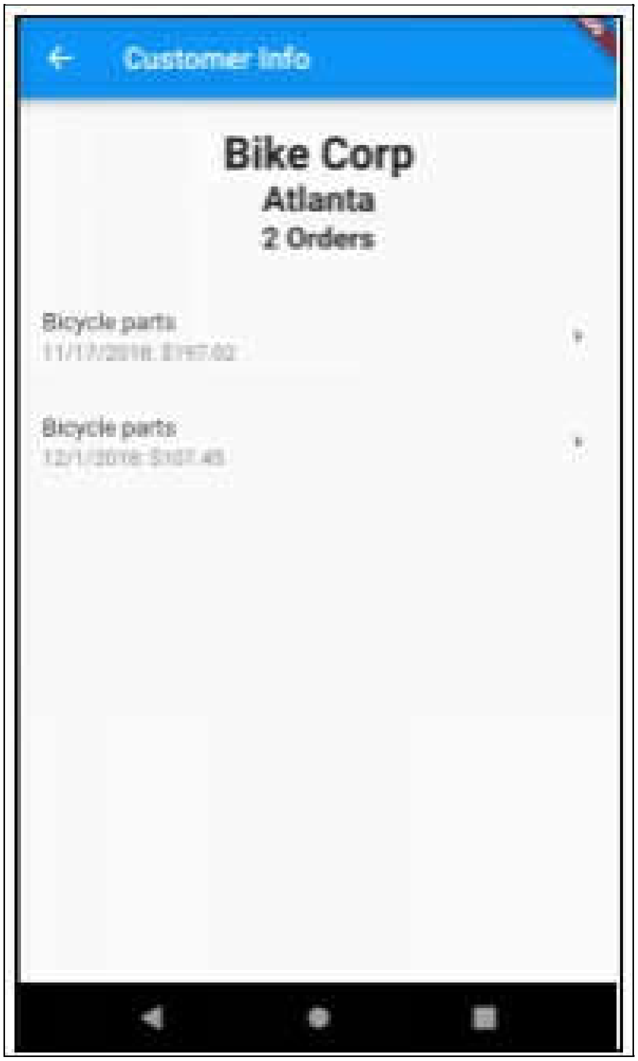
You can Navigate to a Widget and have that Widget return data back to where it was opened. We are not doing this in the Example but it’s good to know you can do this. Take a look at some of the Dialog examples.

If you remember how Dialogs worked, they would close by calling Navigator.pop with a data argument (data to be returned). The ‘push’ method of the Navigator returns a Future, so you can wait for the future to complete to get the data returned from the target Navigation once the user has navigated back.

Example – ‘routes_simple’

This example app allows you to navigate from Customers to Customer Info including Orders to Order Info.


Customers
Bike Corp
Atlanta
Trust Corp
Atlanta
Jilly Boutique
Birmingham


Customer Info
Bike Corp
Atlanta
2 Orders
Bicycle parts
Bicycle parts

Source Code

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

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

class Order {
  DateTime _dt;
  String _description;
  double _total;
```

```
  Order(this._dt, this._description, this._total);

  double get total => _total;
  String get description => _description;
  DateTime get dt => _dt;
}

class Customer {
  String _name;
  String _location;
  List<Order> _orders;

  Customer(this._name, this._location, this._orders);

  List<Order> get orders => _orders;
  String get location => _location;
  String get name => _name;
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
```

```
      home: new HomePageWidget(),
    );
  }
}

class HomePageWidget extends StatelessWidget {
  List<Customer> _customerList = [
    Customer("Bike Corp", "Atlanta", [
      Order(DateTime(2018, 11, 17), "Bicycle parts", 197.02),
      Order(DateTime(2018, 12, 1), "Bicycle parts", 107.45),
    ]),
    Customer("Trust Corp", "Atlanta", [
      Order(DateTime(2017, 1, 3), "Shredder parts", 97.02),
      Order(DateTime(2018, 3, 13), "Shredder blade", 7.45),
      Order(DateTime(2018, 5, 2), "Shredder blade", 7.45),
    ]),
    Customer("Jilly Boutique", "Birmingham", [
      Order(DateTime(2018, 1, 3), "Display unit", 97.01),
      Order(DateTime(2018, 3, 3), "Desk unit", 12.25),
      Order(DateTime(2018, 3, 21), "Clothes rack", 97.15),
    ]),
  ];

  HomePageWidget({Key key}) : super(key: key);

  void navigateToCustomer(BuildContext context, Customer
customer) {
```

```
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) =>
CustomerWidget(customer)),
    );
  }

  ListTile createCustomerWidget(BuildContext context,
Customer customer) {
    return new ListTile(
        title: Text(customer.name),
        subtitle: Text(customer.location),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToCustomer(context, customer));
  }

  @override
  Widget build(BuildContext context) {
    List<Widget> customerList = List.from(_customerList
        .map((Customer customer) =>
createCustomerWidget(context, customer)));
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Customers"),
        ),
        body: new Center(
          child: new ListView(
            children: customerList,
```

```
          ),
        ));
  }
}

class CustomerWidget extends StatelessWidget {
  Customer _customer;

  CustomerWidget(this._customer);

  void navigateToOrder(BuildContext context, Customer
customer, Order order) {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) =>
OrderWidget(customer, order)),
    );
  }

  ListTile createOrderListWidget(
      BuildContext context, Customer customer, Order order) {
    return new ListTile(
        title: Text(order.description),
        subtitle:
Text("${order.dt.month}/${order.dt.day}/${order.dt.year}: "
            "\$${order.total}"),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToOrder(context, customer,
order));
```

```
  }

  @override
  Widget build(BuildContext context) {
    List<Widget> widgetList =
List.from(_customer.orders.map(
        (Order order) => createOrderListWidget(context,
_customer, order)));
    widgetList.insert(
        0,
        Container(
            child: Column(
              children: <Widget>[
                Text(
                  _customer.name,
                  style: TextStyle(fontSize: 30.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  _customer.location,
                  style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  "${_customer.orders.length} Orders",
                  style: TextStyle(fontSize: 20.0, fontWeight:
FontWeight.bold),
                )
```

```
            ],
          ),
          padding: EdgeInsets.all(20.0)));
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Customer Info"),
        ),
        body: new Center(
          child: new ListView(
            children: widgetList,
          ),
        ));
  }
}

class OrderWidget extends StatelessWidget {
  Customer _customer;
  Order _order;

  OrderWidget(this._customer, this._order);

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Order Info"),
        ),
```

```
        body: new Padding(
          padding: EdgeInsets.all(20.0),
          child: new ListView(
            children: <Widget>[
              Text(_customer.name,
                  style: TextStyle(
                    fontSize: 30.0,
                    fontWeight: FontWeight.bold,
                  ),
                  textAlign: TextAlign.center),
              Text(_customer.location,
                  style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center),
              Text(""),
              Text(_order.description,
                  style: TextStyle(fontSize: 18.0, fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center),
              Text(
                  "${_order.dt.month}/${_order.dt.day}/${_order.dt.
"
                  "\$${_order.total}",
                  style: TextStyle(fontSize: 18.0, fontWeight:
FontWeight.bold),
                  textAlign: TextAlign.center)
            ],
```

```
          ),
        ));
  }
}
```

Invoking Navigation with Named Routes #1

Named routes enable us to use routes that are defined just once, avoiding code duplication. These are very easy to use!

Define Routes

We define the routes when we build the MaterialApp at the top of the Widget tree:

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
       ... [other constructor arguments] ...
routes: <String, WidgetBuilder>{
        '/customer': (context) => CustomerWidget(),
        '/order': (context) => OrderWidget(),
      },
    );
  }
```

```
}
```

Navigating Forward

Note how we navigate forward using a name that matches a route defined in the MaterialApp:

```
Navigator.pushNamed(context, "/order");
```

See the problem yet?

The problem is that this approach is great for simple Navigation without passing parameters. It doesn’t work really work when you have parameters. Use this approach only when you have simple Widget navigation.

Example – ‘routes_named’

This app looks and feels the same as the previous example but it does not pass around parameters. It just shows dummy data.

Source Code

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

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

class Order {
  DateTime _dt;
```

```
  String _description;
  double _total;

  Order(this._dt, this._description, this._total);

  double get total => _total;

  String get description => _description;

  DateTime get dt => _dt;
}

class Customer {
  String _name;
  String _location;
  List<Order> _orders;

  Customer(this._name, this._location, this._orders);

  List<Order> get orders => _orders;

  String get location => _location;

  String get name => _name;
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
```

```
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new HomePageWidget(),
      routes: <String, WidgetBuilder>{
        '/customer': (context) => CustomerWidget(), // only
simple routes work
        '/order': (context) => OrderWidget(), // only simple
routes work
      },
    );
  }
}

class HomePageWidget extends StatelessWidget {
  List<Customer> _customerList = [
    Customer("Bike Corp", "Atlanta", []),
    Customer("Trust Corp", "Atlanta", []),
    Customer("Jilly Boutique", "Birmingham", []),
  ];

  HomePageWidget({Key key}) : super(key: key);

  void navigateToCustomer(BuildContext context, Customer
customer) {
    Navigator.pushNamed(context, "/customer"); // only simple
```

```
routes work
  }

  ListTile createCustomerWidget(BuildContext context,
Customer customer) {
    return new ListTile(
        title: Text(customer.name),
        subtitle: Text(customer.location),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToCustomer(context, customer));
  }

  @override
  Widget build(BuildContext context) {
    List<Widget> customerList = List.from(_customerList
        .map((Customer customer) =>
createCustomerWidget(context, customer)));
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Customers"),
        ),
        body: new Center(
          child: new ListView(
            children: customerList,
          ),
        ));
  }
```

```
}

class CustomerWidget extends StatelessWidget {
  List<Order> _orderList = [
    Order(DateTime(2018, 11, 17), "Bicycle parts", 197.00),
    Order(DateTime(2018, 12, 1), "Bicycle parts", 107.45),
  ];

  CustomerWidget({Key key}) : super(key: key);

  void navigateToOrder(BuildContext context, Order order) {
    Navigator.pushNamed(context, "/order"); // only simple
routes work
  }

  ListTile createOrderWidget(BuildContext context, Order
order) {
    return new ListTile(
        title: Text(order.description),
        subtitle:
Text("${order.dt.month}/${order.dt.day}/${order.dt.year}: "
            "\$${order.total}"),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToOrder(context, order));
  }

  @override
  Widget build(BuildContext context) {
    List<Widget> widgetList = List.from(
```

```
        _orderList.map((Order order) =>
createOrderWidget(context, order)));
    widgetList.insert(
        0,
        Container(
            child: Column(
              children: <Widget>[
                Text(
                  "BikeCorp",
                  style: TextStyle(fontSize: 30.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  "Atlanta",
                  style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  "2 Orders",
                  style: TextStyle(fontSize: 20.0, fontWeight:
FontWeight.bold),
                )
              ],
            ),
            padding: EdgeInsets.all(20.0)));
    return new Scaffold(
        appBar: new AppBar(
```

```
        title: new Text("Customers"),
      ),
      body: new Center(
        child: new ListView(
          children: widgetList,
        ),
      ));
  }
}

class OrderWidget extends StatelessWidget {
  OrderWidget();

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text("Order Info"),
      ),
      body: new Padding(
        padding: EdgeInsets.all(20.0),
        child: new ListView(
          children: <Widget>[
            Text("BikeCorp",
                style: TextStyle(
                  fontSize: 30.0,
                  fontWeight: FontWeight.bold,
```

```
              ),
              textAlign: TextAlign.center),
            Text("Atlanta",
              style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
              textAlign: TextAlign.center),
            Text(""),
            Text("Bicycle Parts",
              style: TextStyle(fontSize: 18.0, fontWeight:
FontWeight.bold),
              textAlign: TextAlign.center),
            Text("12/1/2019 \$123.23",
              style: TextStyle(fontSize: 18.0, fontWeight:
FontWeight.bold),
              textAlign: TextAlign.center)
          ],
        ),
      ));
  }
}
```

Invoking Navigation with Named Routes #2

The approach #1 doesn’t work really work when you have parameters and you need to pass data to a route though parameters.

Here is another way of routing with named routes, only this time it works with parameters.

Attach Route Handler to MaterialApp

This time we don't define routes in the Material App. Instead we pass in a route handler to the MaterialApp at the top of the Widget tree (in this case 'handleRoute'):

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      ... [other constructor arguments] ...
onGenerateRoute: handleRoute,
    );
  }
}
```

Code Route Handler

We write a route handler that interprets the route info (parsing out the parameters) and returns a MaterialPageRoute containing a builder to create the correct Widget. This will work to generate the Widgets for all the routing.

Example

In the example below we convert the route info into a MaterialPageRoute for a Customer Widget or an Order Widget. Both receive the id as the constructor argument.

```
Route<dynamic> handleRoute(RouteSettings routeSettings) {
  // One route handler to handle them all.
  List<String> nameParm = routeSettings.name.split(":");
  assert(nameParm.length == 2);
  String name = nameParm[0];
  assert(name != null);
  int id = int.tryParse(nameParm[1]);
  assert(id != null);
  Widget childWidget;
  if (name == "/customer/") {
    childWidget = CustomerWidget(id);
  } else {
    childWidget = OrderWidget(id);
  }
  return MaterialPageRoute(
      builder: (context) => DataContainerWidget(child:
childWidget));
}
```

Navigating Forward

Now we have a route handler that can interpret routes with data, we can route by name and id.

```
void navigateToCustomer(BuildContext context, Customer
customer) {
    Navigator.pushNamed(context,
'/customer/:${customer.id}');
  }
```

Example – ‘routes_named_with_parms’

This app looks and feels the same as the previous example but this time it passes the customer and order identifiers to the Customer and Order Widgets.
I added a DataContainerWidget to store Customer and Order state data in one place (more on InheritedWidgets later in their own chapter) and enable them to be queried by the identifier.
Each widget is constructed (passing in the Customer or Order identifier) then calls code in the DataContainerWidget to get the data to display in the UI.

Source Code

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

void main() => runApp(new MyApp());
```

```
class Order {
  int _id;
  DateTime _dt;
  String _description;
  double _total;

  Order(this._id, this._dt, this._description, this._total);
  Order.empty() : this(0, DateTime.now(), "", 0.0);

  int get id => _id;
  double get total => _total;
  String get description => _description;
  DateTime get dt => _dt;
}

class Customer {
  int _id;
  String _name;
  String _location;
  List<Order> _orders;

  Customer(this._id, this._name, this._location, this._orders);
  Customer.empty() : this(0, "", "", []);

  int get id => _id;
  List<Order> get orders => _orders;
  String get location => _location;
  String get name => _name;
```

```
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
        title: 'Flutter Demo',
        theme: new ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: new DataContainerWidget(child: HomeWidget()),
        onGenerateRoute: handleRoute);
  }

  Route<dynamic> handleRoute(RouteSettings routeSettings) {
    // One route handler to handle them all.
    List<String> nameParm = routeSettings.name.split(":");
    assert(nameParm.length == 2);
    String name = nameParm[0];
    assert(name != null);
    int id = int.tryParse(nameParm[1]);
    assert(id != null);
    Widget childWidget;
    if (name == "/customer/") {
      childWidget = CustomerWidget(id);
    } else {
```

```
      childWidget = OrderWidget(id);
    }
    return MaterialPageRoute(
        builder: (context) => DataContainerWidget(child:
childWidget));
  }
}

class DataContainerWidget extends InheritedWidget {
  DataContainerWidget({
    Key key,
    @required Widget child,
  })  : assert(child != null),
        super(key: key, child: child);

  List<Customer> _customerList = [
    Customer(1, "Bike Corp", "Atlanta", [
      Order(11, DateTime(2018, 11, 17), "Bicycle parts",
197.02),
      Order(12, DateTime(2018, 12, 1), "Bicycle parts",
107.45),
    ]),
    Customer(2, "Trust Corp", "Atlanta", [
      Order(13, DateTime(2017, 1, 3), "Shredder parts", 97.02),
      Order(14, DateTime(2018, 3, 13), "Shredder blade", 7.45),
      Order(15, DateTime(2018, 5, 2), "Shredder blade", 7.45),
    ]),
    Customer(3, "Jilly Boutique", "Birmingham", [
```

```
    Order(16, DateTime(2018, 1, 3), "Display unit", 97.01),
    Order(17, DateTime(2018, 3, 3), "Desk unit", 12.25),
    Order(18, DateTime(2018, 3, 21), "Clothes rack", 97.15),
  ]),
];

List<Customer> get customerList => _customerList;

Customer getCustomer(int id) {
  return _customerList.firstWhere((customer) => customer.id
== id,
      orElse: () => Customer.empty());
}

Customer getCustomerForOrderId(int id) {
  return customerList.firstWhere(
      (customer) => customerHasOrderId(customer, id),
      orElse: () => Customer.empty());
}

Order getOrder(int id) {
  Customer customerThatOwnsOrder =
getCustomerForOrderId(id);
  return customerThatOwnsOrder.orders
      .firstWhere((order) => order.id == id, orElse: () =>
Order.empty());
}

bool customerHasOrderId(Customer customer, int id) {
```

```
    Order order = customer.orders
        .firstWhere((order) => order.id == id, orElse: () =>
Order.empty());
    return order.id != 0;
  }

  static DataContainerWidget of(BuildContext context) {
    return
context.inheritFromWidgetOfExactType(DataContainerWidget)
        as DataContainerWidget;
  }

  @override
  bool updateShouldNotify(covariant InheritedWidget
oldWidget) {
    return false;
  }
}

class HomeWidget extends StatelessWidget {
  HomeWidget({Key key}) : super(key: key);

  void navigateToCustomer(BuildContext context, Customer
customer) {
    Navigator.pushNamed(context,
'/customer/:${customer.id}');
  }

  ListTile createCustomerWidget(BuildContext context,
Customer customer) {
```

```
    return new ListTile(
        title: Text(customer.name),
        subtitle: Text(customer.location),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToCustomer(context, customer));
  }

  @override
  Widget build(BuildContext context) {
    DataContainerWidget data =
DataContainerWidget.of(context);
    List<Widget> customerList = List.from(data.customerList
        .map((Customer customer) =>
createCustomerWidget(context, customer)));
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Customers"),
        ),
        body: new Center(
          child: new ListView(
            children: customerList,
          ),
        ));
  }
}

class CustomerWidget extends StatelessWidget {
```

```
  int _id;

  CustomerWidget(this._id);

  void navigateToOrder(BuildContext context, Order order) {
    Navigator.pushNamed(context, '/order/:${order.id}');
  }

  ListTile createOrderListWidget(BuildContext context, Order
order) {
    return new ListTile(
        title: Text(order.description),
        subtitle:
Text("${order.dt.month}/${order.dt.day}/${order.dt.year}: "
            "\$${order.total}"),
        trailing: Icon(Icons.arrow_right),
        onTap: () => navigateToOrder(context, order));
  }

  @override
  Widget build(BuildContext context) {
    DataContainerWidget data =
DataContainerWidget.of(context);
    Customer customer = data.getCustomer(_id);
    List<Widget> orderListWidgets =
List.from(customer.orders
        .map((Order order) => createOrderListWidget(context,
order)));
    orderListWidgets.insert(
```

```
        0,
        Container(
            child: Column(
              children: <Widget>[
                Text(
                  customer.name,
                  style: TextStyle(fontSize: 30.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  customer.location,
                  style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  "${customer.orders.length} Orders",
                  style: TextStyle(fontSize: 20.0, fontWeight:
FontWeight.bold),
                )
              ],
            ),
            padding: EdgeInsets.all(20.0)));
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Customer Info"),
        ),
        body: new Center(
```

```
        child: new ListView(
          children: orderListWidgets,
        ),
      ));
  }
}

class OrderWidget extends StatelessWidget {
  int _id;

  OrderWidget(this._id);

  @override
  Widget build(BuildContext context) {
    DataContainerWidget data =
        context.inheritFromWidgetOfExactType(DataContainerWi
    Customer customer = data.getCustomerForOrderId(_id);
    Order order = data.getOrder(_id);
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Order Info"),
        ),
        body: new Padding(
          padding: EdgeInsets.all(20.0),
          child: new ListView(
            children: <Widget>[
              Text(customer.name,
```

```
              style: TextStyle(
                fontSize: 30.0,
                fontWeight: FontWeight.bold,
              ),
              textAlign: TextAlign.center),
          Text(customer.location,
              style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),
              textAlign: TextAlign.center),
          Text(""),
          Text(order.description,
              style: TextStyle(fontSize: 18.0, fontWeight: FontWeight.bold),
              textAlign: TextAlign.center),
          Text(
              "${order.dt.month}/${order.dt.day}/${order.dt.year
\$${order.total}",
              style: TextStyle(fontSize: 18.0, fontWeight: FontWeight.bold),
              textAlign: TextAlign.center)
        ],
      ),
    ));
  }
}
```

PageView

Introduction

You can navigate with PageViews as well. PageViews are useful for when you have a list of Widgets that each take up all the screen space and you want to swipe through them, either horizontally or vertically. The 'scrollDirection' constructor argument enables you to set the scrolling / swiping axis to be horizontal or vertical.

Child Widgets

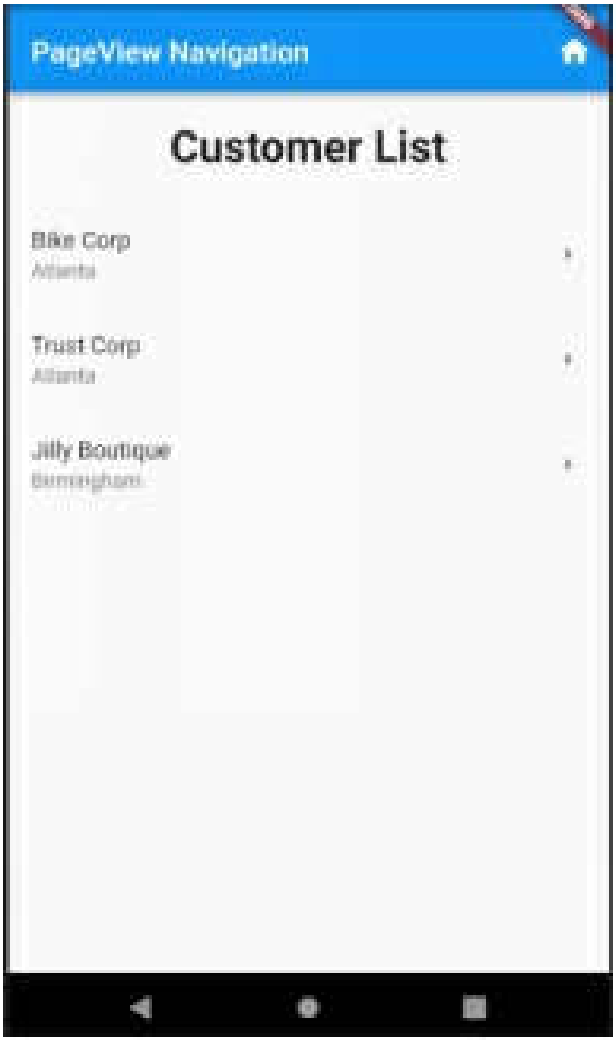
PageViews can work with a list of child Widgets or you can them with a builder that creates child Widgets when they are required. If you want to use a builder then use the 'PageView.builder' named constructor. That is probably much better if you planning on giving the user many pages to swipe through. This Widget uses the 'Page' terminology to refer to a child Widget that takes up all of the available screen space.

Controller

PageViews also work with a controller, which you can specify as an argument in the PageView contructor. You can use the controller to move between the childWidgets. To move between childWidgets with animation, use ‘animateToPage’. To jump to a page without animation, use ‘jumpToPage’. You can also go to the previous and next pages.

Example – ‘page_view_navigation’

This app is similar to the previous apps in this chapter. On the home page, you see a list of customers. You can tap on a customer to move to that Customer’s page, or you can swipe through the Customers. There is a Home button on the toolbar to take you back to the home page.



PageView Navigation
Customer List
Bike Corp
Atlanta
Trust Corp
Atlanta
Jilly Boutique
Birmingham

Source Code

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

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

class Order {
  DateTime _dt;
  String _description;
  double _total;
```

```
  Order(this._dt, this._description, this._total);

  double get total => _total;
  String get description => _description;
  DateTime get dt => _dt;
}

class Customer {
  String _name;
  String _location;
  List<Order> _orders;

  Customer(this._name, this._location, this._orders);

  List<Order> get orders => _orders;
  String get location => _location;
  String get name => _name;
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'PageView Navigation',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
```

```
    home: new MyHomePage(),
   );
  }
}

class MyHomePage extends StatelessWidget {
  final PageController _pageController =
PageController(initialPage: 0);
  final Duration _duration = Duration(seconds: 1);
  final Curve _curve = Curves.ease;

  final List<Customer> _customerList = [
    Customer("Bike Corp", "Atlanta", [
      Order(DateTime(2018, 11, 17), "Bicycle parts", 197.02),
      Order(DateTime(2018, 12, 1), "Bicycle parts", 107.45),
    ]),
    Customer("Trust Corp", "Atlanta", [
      Order(DateTime(2017, 1, 3), "Shredder parts", 97.02),
      Order(DateTime(2018, 3, 13), "Shredder blade", 7.45),
      Order(DateTime(2018, 5, 2), "Shredder blade", 7.45),
    ]),
    Customer("Jilly Boutique", "Birmingham", [
      Order(DateTime(2018, 1, 3), "Display unit", 97.01),
      Order(DateTime(2018, 3, 3), "Desk unit", 12.25),
      Order(DateTime(2018, 3, 21), "Clothes rack", 97.15),
    ]),
  ];
```

```
  MyHomePage({Key key}) : super(key: key);

  Widget pageViewItemBuilder(BuildContext context, int
index) {
    if (index == 0) {
      return createHomePage(context);
    } else {
      return createDetailPage(context, index);
    }
  }

  Widget createHomePage(BuildContext context) {
    List<Widget> widgetList = [];
    widgetList.add(Padding(
        padding: EdgeInsets.all(20.0),
        child: Text(
          "Customer List",
          style: TextStyle(fontSize: 30.0, fontWeight:
FontWeight.bold),
          textAlign: TextAlign.center,
        )));
    for (int i = 0, ii = _customerList.length; i < ii; i++) {
      Customer customer = _customerList[i];
      widgetList.add(createHomePageListItem(context,
customer, i));
    }
    return ListView(children: widgetList);
  }
```

```
  ListTile createHomePageListItem(
      BuildContext context, Customer customer, int index) {
    return new ListTile(
        title: Text(customer.name),
        subtitle: Text(customer.location),
        trailing: Icon(Icons.arrow_right),
        onTap: () => _pageController.animateToPage(index + 1,
            duration: _duration, curve: _curve));
  }

  Widget createDetailPage(BuildContext context, int index) {
    Customer customer = _customerList[index - 1];
    List<Widget> widgetList = List.from(customer.orders
        .map((Order order) => createOrderListWidget(context,
customer, order)));
    widgetList.insert(
        0,
        Container(
            child: Column(
              children: <Widget>[
                Text(
                  customer.name,
                  style: TextStyle(fontSize: 30.0, fontWeight:
FontWeight.bold),
                ),
                Text(
```

```
                  customer.location,
                  style: TextStyle(fontSize: 24.0, fontWeight:
FontWeight.bold),
                ),
                Text(
                  "${customer.orders.length} Orders",
                  style: TextStyle(fontSize: 20.0, fontWeight:
FontWeight.bold),
                ),
                Padding(padding: EdgeInsets.all(20.0)),
              ],
            ),
            padding: EdgeInsets.all(20.0)));
    return ListView(children: widgetList);
  }

  ListTile createOrderListWidget(
      BuildContext context, Customer customer, Order order) {
    return new ListTile(
        title: Text(order.description),
        subtitle:
Text("${order.dt.month}/${order.dt.day}/${order.dt.year}: "
            "\$${order.total}"));
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
```

```
      appBar: new AppBar(title: new Text("PageView
Navigation"), actions: [
        IconButton(
            icon: Icon(Icons.home),
            onPressed: () => _pageController.animateToPage(0,
                duration: _duration, curve: _curve))
      ]),
      body: new Center(
          child: new PageView.builder(
              controller: _pageController,
              itemBuilder: pageViewItemBuilder,
              itemCount: _customerList.length + 1)),
    );
  }
}
```

25. Forms

Introduction

We need to give the users the ability to enter information into forms, fields, validate it and show validation messages to the user if necessary.

The purpose of this chapter is to learn how get Flutter apps working with fields, forms and validations.

Flutter provides objects to help you with the process of building forms, validation and input fields. It provides a Form object, Form Field objects (indirectly) and all the input types below:

- Checkbox
- DropdownButton
- Radio
- Switch
- TextFormField / TextField

Form

This is a Widget that is designed to wrap form Widgets and provides control over validation.

The Form object gives you the following constructor arguments:

- ‘autovalidate’ to enable or disable automatic validation.
- ‘onChanged’ callback fired when one of the fields are changed.

The Form object gives you the following methods:

- ‘reset’ to reset fields.
- ‘save’ to save fields.
- ‘validate’ to validate, returning a true if the form fields are valid, false if one or more are invalid.

Form State

The Form object stores input state data from child TextFormFields but not other field types like Checkboxes, DropdownButtons, Radios, Switches.

So, if you want your form to work with those other types of fields, you need to store the state of those items. If you take a look a look at the example you will see that these fields are stored as state in the Stateful Widget.

Form Validation

As mentioned earlier, the Form class has a ‘autovalidate’ constructor argument.

- If this argument is set to true, the framework invokes validation as data is input.
- If this argument is set to false, the framework will not invoke validation until the ‘validate’ method is invoked.

Form / Field Integration

The FormField is a Widget used as a base class by field Widgets (such as TextFormField) to integrate the field with the parent Form Widget and provide services such as validation.

Form Fields

Checkbox

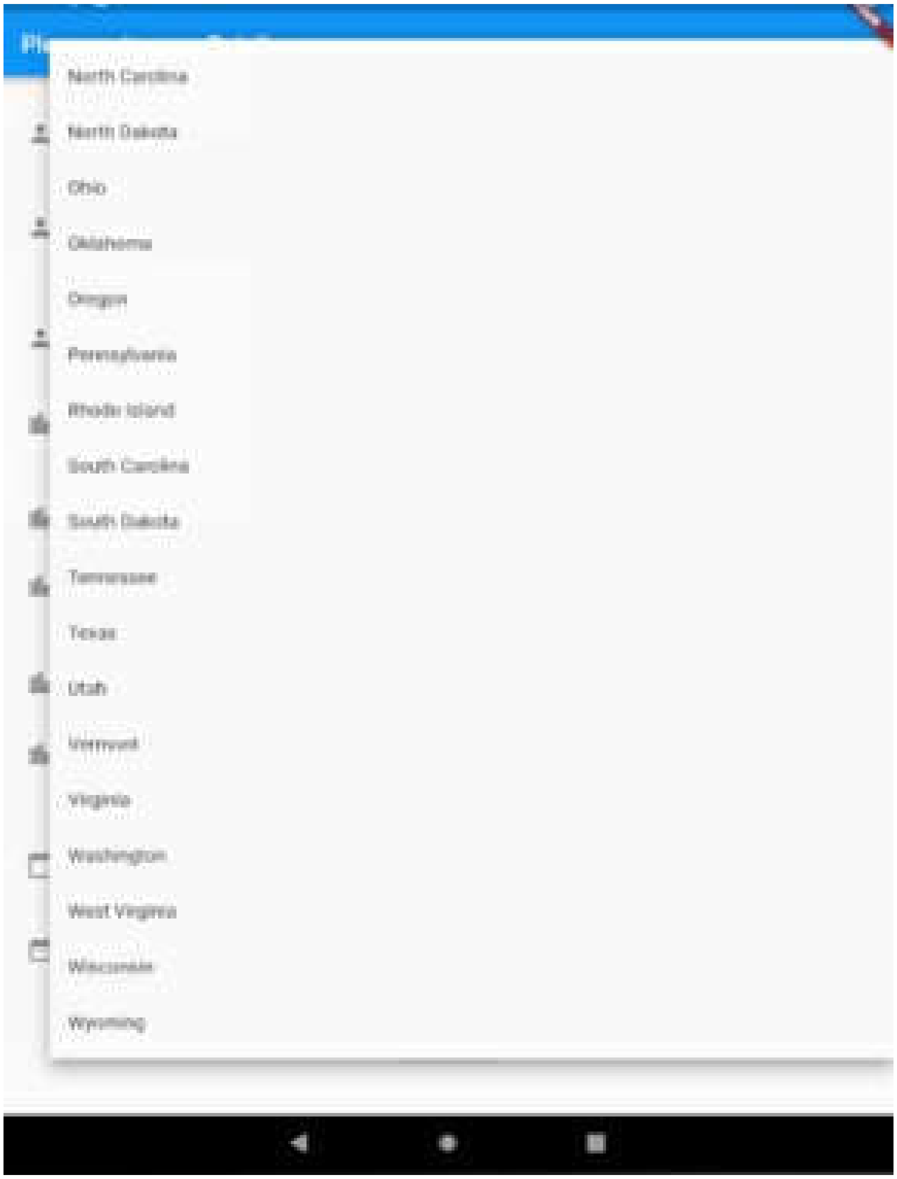
This Widget that allows the user to select a yes / no.

It does not store state for you, you have to manage it yourself.
Use the following constructor arguments for state management.

	Description

value	Sets the value represented by the radio. Provide this from state.
onChanged	Method fired when the checkbox is selected or deselected. Add method to set state here.

DropdownButton



This is a material design button that allows the user to select an item from a list of items that implemented as a popup menu.

It does not store state for you, you have to manage it yourself.
Use the following constructor arguments for state

management.

	Description
items	Sets the items in the list.
value	Sets the currently selected item. Provide this from state.
onChanged	Method fired when an item is selected or deselected. Add method to set state here.

Radio

This Widget does not store state for you, you have to manage it yourself.
Use the following constructor arguments for state management.

	Description
value	Sets the value represented by the radio.
groupValue	Sets the radio button’s value. Provide this from state.
onChanged	Method fired when the radio button is selected. Add method to set state here.

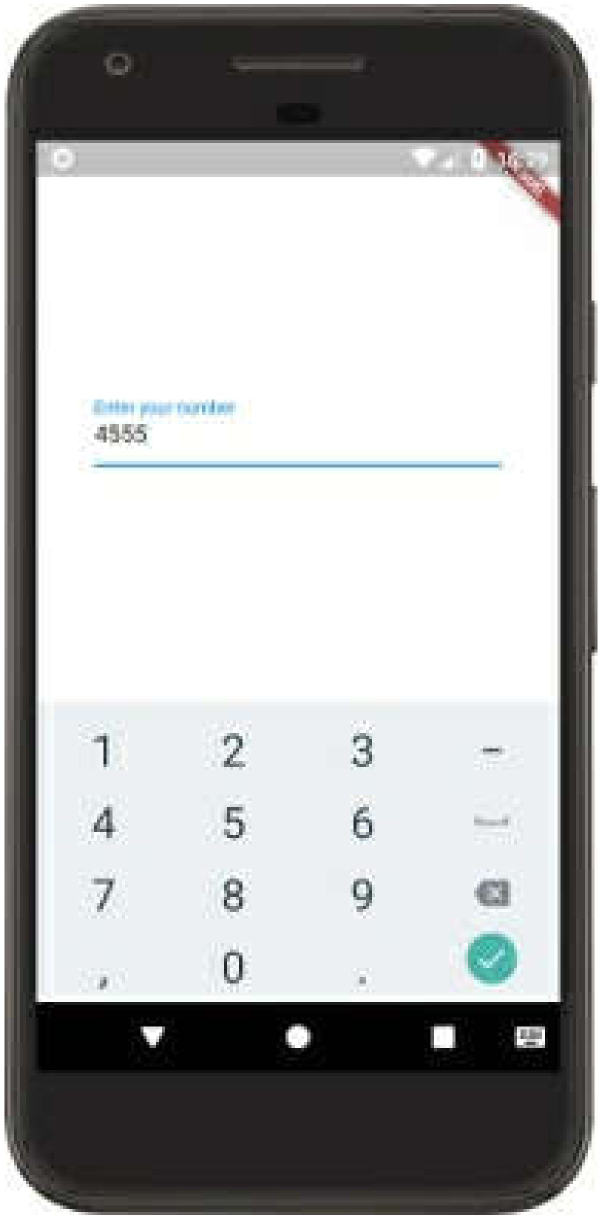
TextFormField, TextField

A TextField is a widget for a basic text field.
A TextFormField is a TextField with form integration.

Keyboard Types

The TextFormField object has a constructor argument ‘keyboardType’. This lets you change the keyboard type to suit the field:

	Description
TextInputType.text	Default keyboard.
TextInputType.multiline	Default keyboard optimized for multiline entry.
TextInputType.number	Numeric keyboard.
TextInputType.phone	Phone keyboard.



InputFormatters

The TextFormField object has a constructor argument ‘inputFormatters’. This lets you change the behavior of the field – what characters this input field will accept.

LengthLimitingTextInputFormatter
WhitelistingTextInputFormatter.digitsOnly
BlacklistingTextInputFormatter.singleLineFormatt
WhitelistingTextInputFormatter
BlacklistingTextInputFormatter

TextEditingController

A TextEditingController is a class that listens to its assigned TextField, and updates its own internal state

every time the text in the TextField changes. Listeners can then read the text and selection properties to learn what the user has typed or how the selection has been updated.
If you look at the example code you will see a TextEditingController for each TextFormField. These TextEditingControllers are used to get and set the values for these fields.

Validator

The TextFormField object has a constructor argument 'validator'. This lets you add a validation method to the field. If there is an error with the information the user has provided, the validator method must return a String containing an error message. If there are no errors, the method should not return anything.

Example

```
TextFormField(
  // The validator receives the text the user has typed in
  validator: (value) {
    if (value.isEmpty) {
      return 'Please enter some text';
    }
  },
);
```

Focus

TextFormFields also have a constructor argument that called ‘autofocus’ that sets up the text field to automatically be the first one with the focus. The other fields like Checkboxes, DropdownButtons, Radios don’t have this.

InputDecorator

Input Decorators are widgets are used to decorate our fields, to give them things like:

- Icon
- Hint
- Label

Changing Error Formatting

Input Decorators default to showing errors in red. However they have many constructor arguments that enable you to customize how your errors look.

Constants File

```
const INPUT_DECORATION_ERROR_STYLE = const
TextStyle(
  color: [whatever],
  fontFamily: [whatever],
  fontSize: [whatever],
```

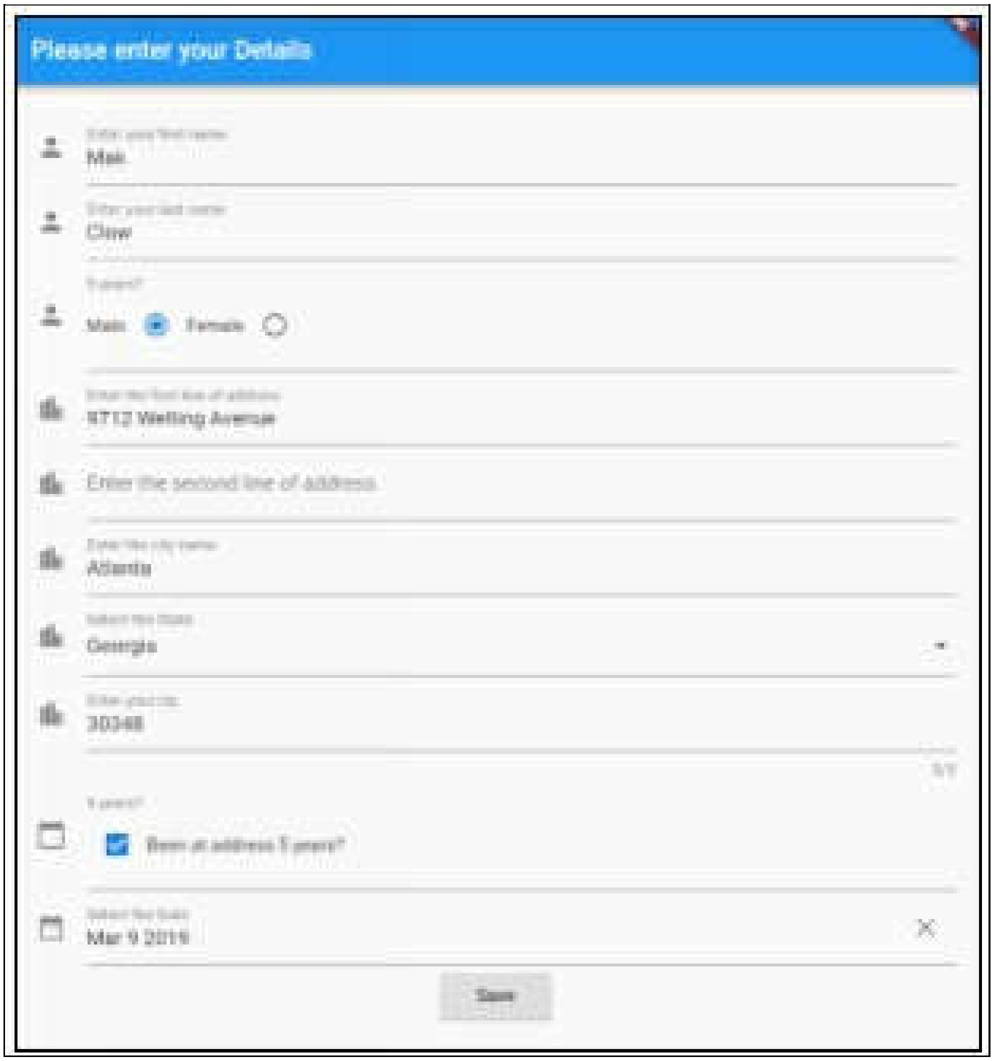
```
  fontWeight: [whatever]
);

const INPUT_DECORATION_ERROR_BORDER = const
UnderlineInputBorder(
    borderSide: BorderSide(color: [whatever])
);

const INPUT_DECORATION_FOCUSED_ERROR_BORDER =
const UnderlineInputBorder(
    borderSide: BorderSide(width: [whatever], color:
[whatever])
);
```

Widget Class

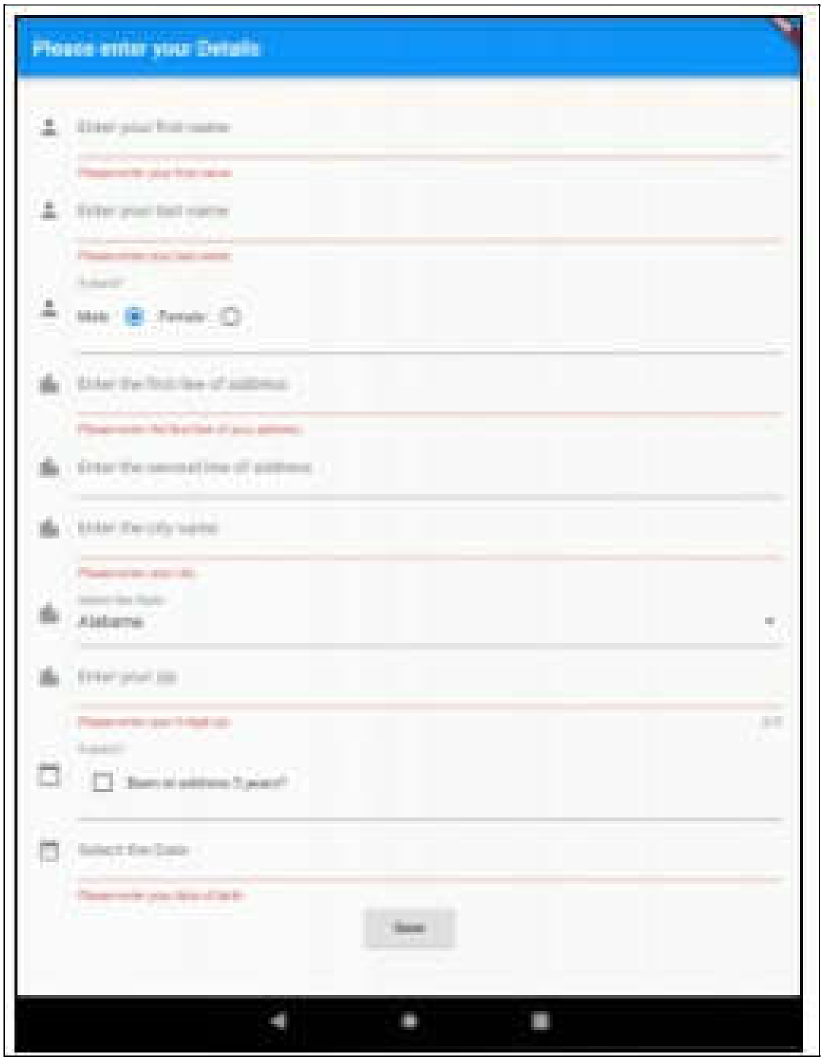
```
InputDecoration(hintText: hint, labelText: label,
        errorStyle: INPUT_DECORATION_ERROR_STYLE,
        errorBorder: INPUT_DECORATION_ERROR_BORDER,
        focusedErrorBorder:
INPUT_DECORATION_FOCUSED_ERROR_BORDER);
```

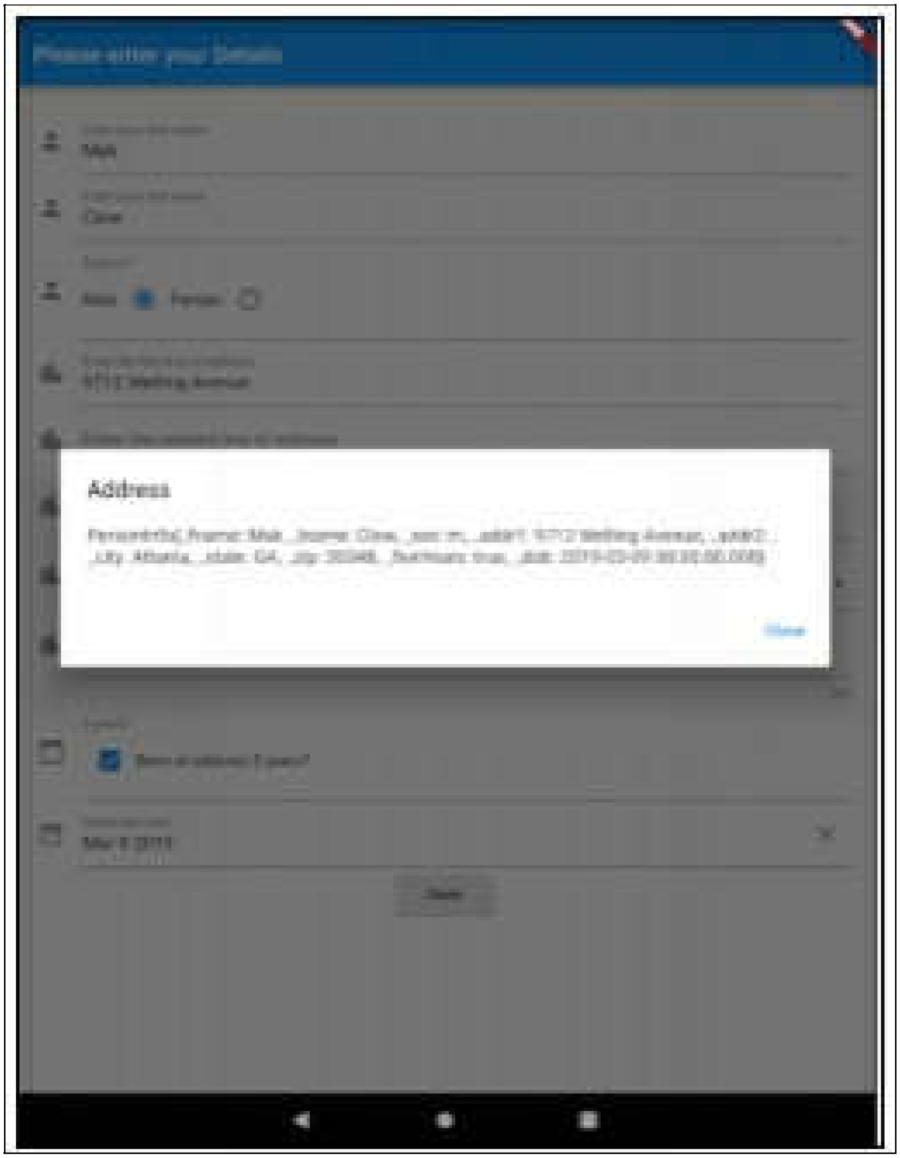
Example – ‘form_details’

This example attempts to use all the input field types: text, radio buttons, checkboxes, selection lists and dates.



Please enter your Details
Max
Clow
Male
Female
9712 Welling Avenue
Enter the second line of address
Atlanta
Georgia
30348
Been at address 5 years?
Mar 9 2019
Save


Please enter your Details
Male
Female
Alabama
Save



Dependencies

Add the following dependencies to your ‘pubspec.yaml’ file. After that you will need to do a ‘flutter packages get’ on the command line in the root of your project to download the dependencies.

```
dependencies:
```

```
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your
application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2
  datetime_picker_formfield: ^0.1.3
```

Source Code

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import
'package:datetime_picker_formfield/datetime_picker_formfield.
import 'package:intl/intl.dart';

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

class PersonInfo {
  String _fname = "";
  String _lname = "";
  String _sex = "m";
  String _addr1 = "";
  String _addr2 = "";
  String _city = "";
  String _state = "";
  String _zip = "";
  bool _fiveYears = false;
```

```
  DateTime _dob;

  PersonInfo(this._fname, this._lname, this._sex, this._addr1, this._addr2,
      this._city, this._state, this._zip, this._fiveYears, this._dob);

  PersonInfo.empty();

  String get fname => _fname;
  String get lname => _lname;
  String get sex => _sex;
  String get addr1 => _addr1;
  String get addr2 => _addr2;
  String get city => _city;
  String get state => _state;
  String get zip => _zip;
  bool get fiveYears => _fiveYears;
  DateTime get dob => _dob;

  @override
  String toString() {
    return 'PersonInfo{_fname: $_fname, _lname: $_lname, _sex: $_sex, _addr1: $_addr1, _addr2: $_addr2, _city: $_city, _state: $_state, _zip: $_zip, _fiveYears: $_fiveYears, _dob: $_dob}';
  }
}
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  PersonInfo _address = PersonInfo.empty();

  HomePage({Key key}) : super(key: key);

  @override
  _HomePageState createState() => new
_HomePageState(_address);
}

class _HomePageState extends State<HomePage> {
  PersonInfo _address;

  _HomePageState(this._address);

  @override
```

```
 Widget build(BuildContext context) {
   return new Scaffold(
       appBar: new AppBar(
         title: new Text("Please enter your Details"),
       ),
       body: new Center(
           child: new ListView(children: [
         Padding(
             padding: EdgeInsets.all(20.0),
             child: AddressWidget(address: _address, onSaved:
_onSaved))
       ])));
 }

 _onSaved(PersonInfo address) {
   showDialog<bool>(
       context: context,
       builder: (BuildContext context) {
         return AlertDialog(
           title: const Text('Address'),
           content: Text(address.toString()),
           actions: <Widget>[
             FlatButton(
               onPressed: () {
                 Navigator.pop(context, true);
               },
```

```
              child: const Text('Close'),
            )
          ],
        );
      });
  }
}

class AddressWidget extends StatefulWidget {
  PersonInfo _address;
  ValueChanged<PersonInfo> _onSaved;

  AddressWidget(
      {Key key,
      @required PersonInfo address,
      @required ValueChanged<PersonInfo> onSaved})
      : super(key: key) {
    this._address = address;
    this._onSaved = onSaved;
  }

  @override
  _AddressWidgetState createState() => new
_AddressWidgetState(_address);
}

class _AddressWidgetState extends State<AddressWidget> {
  static const STATE_DROPDOWN_MENU_ITEMS = [
```

```
DropdownMenuItem(value: "AL", child: const
Text("Alabama")),
DropdownMenuItem(value: "AK", child: const
Text("Alaska")),
DropdownMenuItem(value: "AZ", child: const
Text("Arizona")),
DropdownMenuItem(value: "AR", child: const
Text("Arkansas")),
DropdownMenuItem(value: "CA", child: const
Text("California")),
DropdownMenuItem(value: "CO", child: const
Text("Colorado")),
DropdownMenuItem(value: "CT", child: const
Text("Connecticut")),
DropdownMenuItem(value: "DE", child: const
Text("Delaware")),
DropdownMenuItem(value: "DC", child: const
Text("District Of Columbia")),
DropdownMenuItem(value: "FL", child: const
Text("Florida")),
DropdownMenuItem(value: "GA", child: const
Text("Georgia")),
DropdownMenuItem(value: "HI", child: const
Text("Hawaii")),
DropdownMenuItem(value: "ID", child: const
Text("Idaho")),
DropdownMenuItem(value: "IL", child: const
Text("Illinois")),
DropdownMenuItem(value: "IN", child: const
Text("Indiana")),
DropdownMenuItem(value: "IA", child: const
```

```
Text("Iowa")),
    DropdownMenuItem(value: "KS", child: const
Text("Kansas")),
    DropdownMenuItem(value: "KY", child: const
Text("Kentucky")),
    DropdownMenuItem(value: "LA", child: const
Text("Louisiana")),
    DropdownMenuItem(value: "ME", child: const
Text("Maine")),
    DropdownMenuItem(value: "MD", child: const
Text("Maryland")),
    DropdownMenuItem(value: "MA", child: const
Text("Massachusetts")),
    DropdownMenuItem(value: "MI", child: const
Text("Michigan")),
    DropdownMenuItem(value: "MN", child: const
Text("Minnesota")),
    DropdownMenuItem(value: "MS", child: const
Text("Mississippi")),
    DropdownMenuItem(value: "MO", child: const
Text("Missouri")),
    DropdownMenuItem(value: "MT", child: const
Text("Montana")),
    DropdownMenuItem(value: "NE", child: const
Text("Nebraska")),
    DropdownMenuItem(value: "NV", child: const
Text("Nevada")),
    DropdownMenuItem(value: "NH", child: const Text("New
Hampshire")),
    DropdownMenuItem(value: "NJ", child: const Text("New
Jersey")),
```

```
    DropdownMenuItem(value: "NM", child: const Text("New Mexico")),
    DropdownMenuItem(value: "NY", child: const Text("New York")),
    DropdownMenuItem(value: "NC", child: const Text("North Carolina")),
    DropdownMenuItem(value: "ND", child: const Text("North Dakota")),
    DropdownMenuItem(value: "OH", child: const Text("Ohio")),
    DropdownMenuItem(value: "OK", child: const Text("Oklahoma")),
    DropdownMenuItem(value: "OR", child: const Text("Oregon")),
    DropdownMenuItem(value: "PA", child: const Text("Pennsylvania")),
    DropdownMenuItem(value: "RI", child: const Text("Rhode Island")),
    DropdownMenuItem(value: "SC", child: const Text("South Carolina")),
    DropdownMenuItem(value: "SD", child: const Text("South Dakota")),
    DropdownMenuItem(value: "TN", child: const Text("Tennessee")),
    DropdownMenuItem(value: "TX", child: const Text("Texas")),
    DropdownMenuItem(value: "UT", child: const Text("Utah")),
    DropdownMenuItem(value: "VT", child: const Text("Vermont")),
```

```
    DropdownMenuItem(value: "VA", child: const
Text("Virginia")),
    DropdownMenuItem(value: "WA", child: const
Text("Washington")),
    DropdownMenuItem(value: "WV", child: const Text("West
Virginia")),
    DropdownMenuItem(value: "WI", child: const
Text("Wisconsin")),
    DropdownMenuItem(value: "WY", child: const
Text("Wyoming"))
  ];

  final _formKey = GlobalKey<FormState>();
  String _state =
STATE_DROPDOWN_MENU_ITEMS[0].value;
  TextEditingController _fnameTextController;
  TextEditingController _lnameTextController;
  String _sex = "m";
  TextEditingController _addr1TextController;
  TextEditingController _addr2TextController;
  TextEditingController _cityTextController;
  TextEditingController _zipTextController;
  bool _fiveYears = false;
  DateFormat _dateFormat = DateFormat("MMM d yyyy");
  TextEditingController _dobTextController;

  _AddressWidgetState(final PersonInfo address) {
    _fnameTextController = TextEditingController(text:
address.fname);
```

```
    _lnameTextController = TextEditingController(text:
address.lname);
    _sex = address.sex;
    _addr1TextController = TextEditingController(text:
address.addr1);
    _addr2TextController = TextEditingController(text:
address.addr2);
    _cityTextController = TextEditingController(text:
address.city);
    _zipTextController = TextEditingController(text:
address.state);
    _fiveYears = address.fiveYears;
    _dobTextController = TextEditingController(
        text: address.dob != null ?
_dateFormat.format(address.dob) : "");
  }

  @override
  Widget build(BuildContext context) {
    List<Widget> formWidgetList = new List();
    formWidgetList.add(createFNameWidget());
    formWidgetList.add(createLNameWidget());
    formWidgetList.add(createSexWidget());
    formWidgetList.add(createAddr1Widget());
    formWidgetList.add(createAddr2Widget());
    formWidgetList.add(createCityWidget());
    formWidgetList.add(createStateWidget());
    formWidgetList.add(createZipWidget());
```

```
    formWidgetList.add(createFiveYearsWidget());
    formWidgetList.add(createDobWidget());
    formWidgetList.add(RaisedButton(
      onPressed: () {
        if (_formKey.currentState.validate()) {
          PersonInfo address =
createDataObjectFromFormData();
          widget._onSaved(address);
        }
      },
      child: new Text('Save'),
    ));

    return Form(key: _formKey, child: Column(children:
formWidgetList));
  }

  TextFormField createFNameWidget() {
    return new TextFormField(
        validator: (value) {
          if (value.isEmpty) {
            return 'Please enter your first name.';
          }
        },
        decoration: InputDecoration(
            icon: const Icon(Icons.person),
            hintText: 'First name',
```

```
        labelText: 'Enter your first name'),
      onSaved: (String value) {},
      controller: _fnameTextController,
      autofocus: true);
}

TextFormField createLNameWidget() {
  return new TextFormField(
      validator: (value) {
        if (value.isEmpty) {
          return 'Please enter your last name.';
        }
      },
      decoration: InputDecoration(
          icon: const Icon(Icons.person),
          hintText: 'Last name',
          labelText: 'Enter your last name'),
      onSaved: (String value) {},
      controller: _lnameTextController);
}

void _handleSexRadioChanged(String value) {
  setState(() {
    _sex = value;
  });
}
```

```
InputDecorator createSexWidget() {
  List<Widget> radioWidgets = [
    Text("Male"),
    Radio(
        value: "m",
        groupValue: _sex,
        onChanged: (s) => _handleSexRadioChanged(s)),
    Text("Female"),
    Radio(
        value: "f",
        groupValue: _sex,
        onChanged: (s) => _handleSexRadioChanged(s)),
  ];
  return InputDecorator(
      decoration: const InputDecoration(
        icon: const Icon(Icons.person),
        hintText: 'Been at address 5 years?',
        labelText: '5 years?',
      ),
      child: new DropdownButtonHideUnderline(
          child: Row(children: radioWidgets)));
}

TextFormField createAddr1Widget() {
  return new TextFormField(
      validator: (value) {
```

```
        if (value.isEmpty) {
          return 'Please enter the first line of your address.';
        }
      },
      decoration: InputDecoration(
          icon: const Icon(Icons.location_city),
          hintText: 'Address 1',
          labelText: 'Enter the first line of address'),
      onSaved: (String value) {},
      controller: _addr1TextController);
}

TextFormField createAddr2Widget() {
  return new TextFormField(
      decoration: InputDecoration(
          icon: const Icon(Icons.location_city),
          hintText: 'Address 2',
          labelText: 'Enter the second line of address'),
      onSaved: (String value) {},
      controller: _addr2TextController);
}

TextFormField createCityWidget() {
  return new TextFormField(
      validator: (value) {
        if (value.isEmpty) {
          return 'Please enter your city.';
```

```
        }
      },
      decoration: InputDecoration(
          icon: const Icon(Icons.location_city),
          hintText: 'City',
          labelText: 'Enter the city name'),
      onSaved: (String value) {},
      controller: _cityTextController);
  }

  InputDecorator createStateWidget() {
    DropdownButton<String> stateDropdownButton =
DropdownButton<String>(
      items: STATE_DROPDOWN_MENU_ITEMS,
      value: _state,
      isDense: true,
      onChanged: (String value) {
        setState(() {
          this._state = value;
        });
      });
    return InputDecorator(
      decoration: const InputDecoration(
        icon: const Icon(Icons.location_city),
        hintText: 'Select the State',
        labelText: 'Select the State',
      ),
```

```
        child: new DropdownButtonHideUnderline(child:
stateDropdownButton));
  }

  TextFormField createZipWidget() {
    return new TextFormField(
        validator: (value) {
          if ((value.isEmpty) || (value.length < 5)) {
            return 'Please enter your 5 digit zip.';
          }
        },
        maxLength: 5,
        maxLengthEnforced: true,
        keyboardType: TextInputType.phone,
        inputFormatters:
[WhitelistingTextInputFormatter.digitsOnly],
        decoration: InputDecoration(
            icon: const Icon(Icons.location_city),
            hintText: 'Zip',
            labelText: 'Enter your zip'),
        onSaved: (String value) {},
        controller: _zipTextController);
  }

  InputDecorator createFiveYearsWidget() {
    Checkbox fiveYearsCheckbox = Checkbox(
        value: this._fiveYears,
```

```
      onChanged: (value) {
        setState(() {
          this._fiveYears = value;
        });
      });
  return InputDecorator(
      decoration: const InputDecoration(
        icon: const Icon(Icons.calendar_today),
        hintText: 'Been at address 5 years?',
        labelText: '5 years?',
      ),
      child: new DropdownButtonHideUnderline(
          child: Row(children: [
        fiveYearsCheckbox,
        Text("Been at address 5 years?")
      ])));
}

DateTimePickerFormField createDobWidget() {
  return new DateTimePickerFormField(
      validator: (value) {
        if ((value == null)) {
          return 'Please enter your date of birth.';
        }
      },
      dateOnly: true,
```

```
      format: _dateFormat,
      decoration: InputDecoration(
          icon: const Icon(Icons.date_range),
          hintText: 'Date',
          labelText: 'Select the Date'),
      controller: _dobTextController);
  }

  PersonInfo createDataObjectFromFormData() {
    return new PersonInfo(
      _fnameTextController.text,
      _lnameTextController.text,
      _sex,
      _addr1TextController.text,
      _addr2TextController.text,
      _cityTextController.text,
      _state,
      _zipTextController.text,
      _fiveYears,
      _dateFormat.parse(_dobTextController.text));
  }
}
```

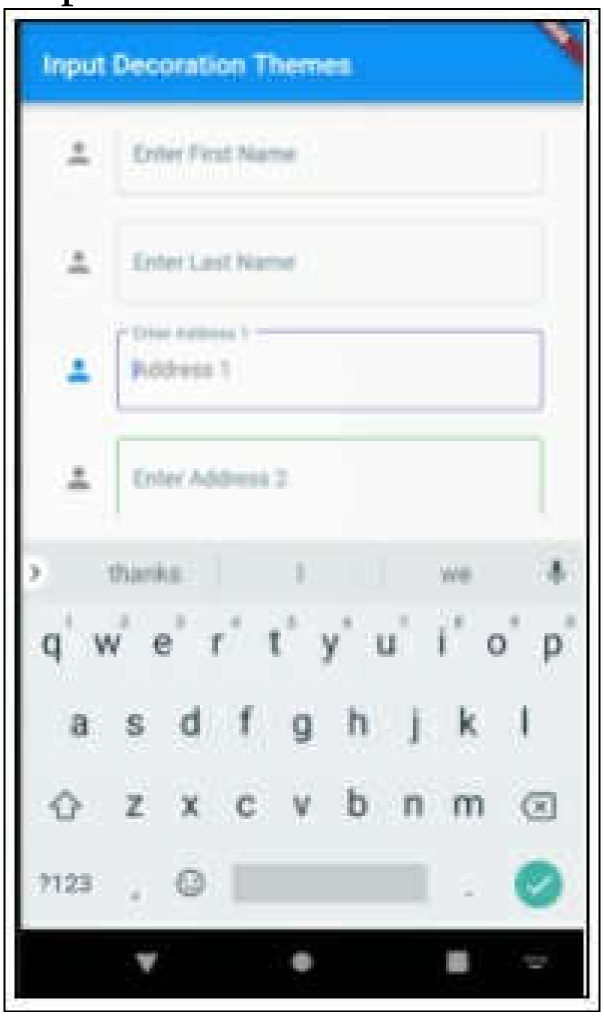
Other Information

Input Decoration Themes

If you don’t like the way the forms look or if you feel they don’t highlight the field states well enough, you can change them in the theme.

Example - ‘input_decoration_themes’

This app shows how your theme can change the appearance of input fields.



Source Code

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

```
void main() => runApp(new MyApp());

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
        inputDecorationTheme: InputDecorationTheme(
          border: const OutlineInputBorder(
            borderSide: BorderSide(color: Colors.blueGrey),
          ),
          enabledBorder: OutlineInputBorder(
            borderSide: BorderSide(color: Colors.green),
          ),
          focusedBorder: const OutlineInputBorder(
            borderSide: BorderSide(color: Colors.deepPurple),
          ),
          labelStyle: const TextStyle(
            color: Colors.blueGrey,
          ),
        ),
      ),
      home: new HomeWidget(),
```

```
  );
 }
}

class HomeWidget extends StatelessWidget {
 final _formKey = GlobalKey<FormState>();
 List<TextEditingController> _textEditingControllers = [];
 List<Widget> _widgets = [];

 HomeWidget({Key key}) : super(key: key) {
  List<String> fieldNames = [
   "First Name",
   "Last Name",
   "Address 1",
   "Address 2",
   "City",
   "State",
   "Zip"
  ];
  for (int i = 0, ii = fieldNames.length; i < ii; i++) {
   String fieldName = fieldNames[i];
   TextEditingController textEditingController =
     new TextEditingController(text: "");
   _textEditingControllers.add(textEditingController);
   _widgets.add(Padding(
    child: _createTextFormField(fieldName, i > 1,
textEditingController),
```

```
        padding: EdgeInsets.all(10.0),
      ));
    }
    _widgets.add(RaisedButton(
      onPressed: () {
        _formKey.currentState.validate();
      },
      child: new Text('Save'),
    ));
  }

  TextFormField _createTextFormField(
      String fieldName, bool enabled, TextEditingController
controller) {
    return new TextFormField(
        enabled: enabled,
        validator: (value) {
          if (value.isEmpty) {
            return 'Please enter ${fieldName}.';
          }
        },
        decoration: InputDecoration(
            icon: const Icon(Icons.person),
            hintText: fieldName,
            labelText: 'Enter ${fieldName}'),
        controller: controller);
  }
```

```
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: new AppBar(
          title: new Text("Input Decoration Themes"),
        ),
        body: Padding(
            padding: EdgeInsets.all(20.0),
            child: Form(
                key: _formKey,
                child: ListView(
                  children: _widgets,
                ))));
  }
}
```

Enabling / Disabling Form Buttons

When dealing with forms, remember that you can enable or disable buttons using the ‘onPressed’ constructor argument:

- If this argument is non-null, then the button is enabled.
- If this argument is null then the button is disabked.

Example – ‘button_enablement’

This app only enables the register button when the user checks the checkbox to agree to the agreement.

Source Code

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
```

```
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new HomeWidget(title: 'Button Enablement'),
    );
  }
}

class HomeWidget extends StatefulWidget {
  HomeWidget({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _HomeWidgetState createState() => new
_HomeWidgetState();
}

class _HomeWidgetState extends State<HomeWidget> {
  bool _checked = false;

  void _onCheck(val) {
    setState(() {
      _checked = val;
    });
  }

  void _onSubmit() {
```

```
    debugPrint("_onSubmit");
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: new AppBar(
          title: new Text(widget.title),
        ),
        body: new Center(
          child: new Column(
            mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
            children: <Widget>[
              new Text('Please check below to agree to the terms.',
                  style: const TextStyle(fontStyle:
FontStyle.italic)),
              Row(mainAxisAlignment:
MainAxisAlignment.center, children: [
                Checkbox(value: _checked, onChanged: (val) =>
_onCheck(val)),
                Text("I agree")
              ]),
              OutlineButton(
                onPressed: _checked ? () => _onSubmit() : null,
                child: const Text('Register'),
              )
```

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