Extracted from:

Programming Flutter

Native, Cross-Platform Apps the Easy Way

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Programming Flutter

Native, Cross-Platform Apps the Easy Way

Carmine Zaccagnino
edited by Michael Swaine

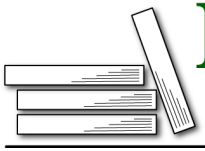
Programming Flutter

Native, Cross-Platform Apps the Easy Way

Carmine Zaccagnino

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Michael Swaine
Copy Editor: Jasmine Kwityn
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-695-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—February 2020

What You Need to Build the UI: Navigation and the InheritedWidget

Before we actually build the UI for this app, we're going to discuss two important things: how to navigate between multiple views in Flutter and how to use InheritedWidgets. This will be a general introduction to the options Flutter offers for navigation; we'll only use some of these in the app. If you'd prefer to jump ahead to more advanced navigation, take a look at Build the App's Basic UI, on page ? and then come back here when you want to learn about how to implement navigation in Flutter apps.

Navigation in Flutter

Flutter offers multiple options for navigating between views:

- push/pop navigation, useful when there is a home page and many pages accessible from it, potentially using a Drawer menu.
- Page-by-page navigation using the PageView and PageController, useful when there are a limited number of pages that can be considered on the same level.

Push/Pop Navigation Using the Navigator

Making an app that takes advantage of the Navigator for push/pop navigation is really simple.

Define the Pages

The first thing to do is to define the pages, which will be really simple:

```
class NewPage extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("New Page"),),
      body: Center(
        child: FlatButton(
          color: Colors.black12,
          onPressed: () {},
          child: Text("Go back to the home page"),
        )
      ),
    );
  }
}
```

```
class HomePage extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Home Page"),),
      body: Center(
        child: Container(
          height: 100,
          child: Card(
            child: Column(
              children: <Widget>[
                Padding(
                  padding: EdgeInsets.all(10.0),
                  child: Text(
                    "Click the button to go to a new page",
                    style: Theme.of(context).textTheme.title,
                  ),
                ),
                ButtonTheme.bar(
                  child: FlatButton(
                    child: Text("Go to new page"),
                    onPressed: () {}
                  ),
                )
              ],
            )
          ),
        ),
      ),
    );
  }
}
```

which generates the following app in its initial state as shown in the screenshot on page 7.

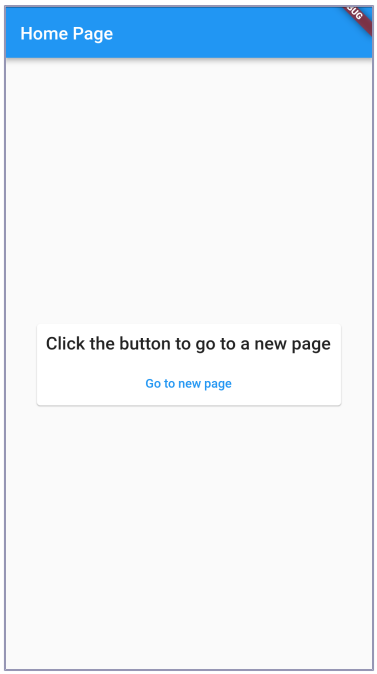
I made the home page a bit more complex than it needed to be to differentiate the two pages visually.

Now comes the important part: that onPressed option for both pages, inside which we are going to add the navigation code.

I called this type of navigation &lquot;push&rquot;/&lquot;pop&rquot; because it uses two methods called Navigator.push() and Navigator.pop().

Push

You can navigate to a new page (contained in a Widget called NewPage) using the Navigator.push() in the following way:

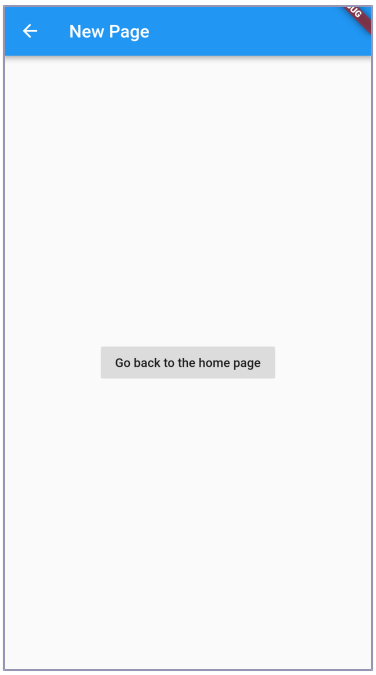



```
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => NewPage(),
  ),
);
```

Navigator.push() takes two arguments: the BuildContext and a Route to the new page.

You can create a route using the MaterialPageRoute() constructor, which uses a builder callback function (which can be a call to the widget's constructor, as is the case in the example) to build the new view.

You can put that code inside the curly braces of the onPressed option for the FlatButton inside MyHomePage and the following view will be displayed when the user taps the button:



Pop

Even though the framework by itself has added a (working, by the way) *back* button in the top left, it is useful to know how to make a custom button to go back.

It is also incredibly simple:

```
Navigator.pop(context);
```

Adding that code to the curly braces of the `onPressed` attribute of `NewPage`'s `FlatButton`, tapping that button in the middle of the screen will make the app go back to the home page.

Using a Drawer Side Navigation Menu

A commonly seen navigation element is the `Drawer`,[5] which is the menu, seen in many apps, that opens by swiping right starting from the left edge of the screen or by tapping the commonly seen button with three vertically stacked horizontal lines.

5. https://material.io/design/components/navigation-drawer.html

Like many items that are commonly used and placed outside the app body, a drawer is added to the app by using the drawer option of the Scaffold and setting it to a Drawer object:

```
Scaffold(
  appBar: AppBar(title: ...),
  drawer: Drawer(
    child: ...
  ),
  body: ...,
)
```

The Drawer class takes just three arguments (excluding the Key), which makes it one of the simplest widgets in the entire Flutter standard library in this respect.

Since one of them is the semanticLabel (which is a string used by accessibility tools such as screen readers to describe what they're seeing) and the other is the elevation (which is a double, also present in the Card, which is used to customize the shadow behind it), the only one that we're currently interested in is the child, which is usually a Column or a ListView (if you're worried about items not having enough vertical space).

For this example, we'll use a ListView.

Very often, the first element in the list is a DrawerHeader, which is a lot like a Container and usually includes either account information and facilities to switch accounts (if present), the name of the app and/or information about its creators, or a short description of what the menu is for.

Below it, a number of ListTiles are used to display navigation options.

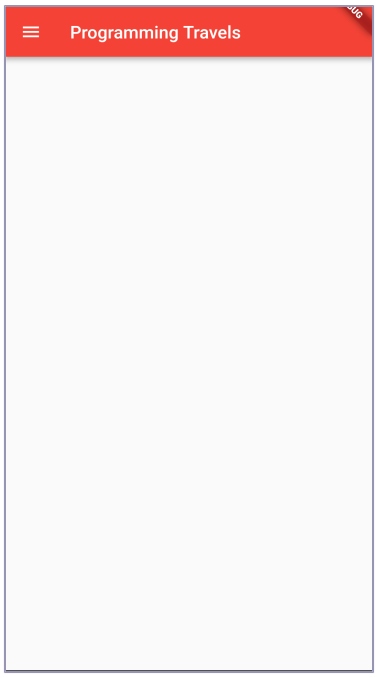
If you decide to use a ListView, you should set its padding option to EdgeInsets.zero to avoid having gaps at the top and bottom of the screen.

Since we're talking about navigation so much, a Drawer for a public transportation or travel app could be the following:

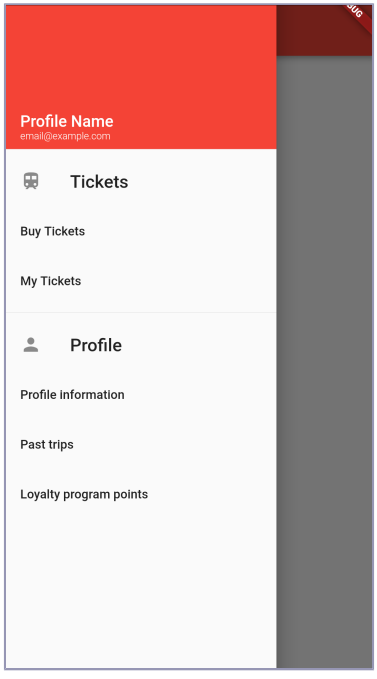
```
Drawer(
  child: ListView(
    padding: EdgeInsets.zero,
    children: [
      DrawerHeader(
        decoration: BoxDecoration(
          color: Colors.red,
        ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          mainAxisAlignment: MainAxisAlignment.end,
```

```
          children: [
            Text(
              "Profile Name",
              style: TextStyle(
                color: Colors.white,
                fontSize: 18,
              ),
            ),
            Text(
              "email@example.com",
              style: TextStyle(
                color: Colors.white,
                fontSize: 11,
                fontWeight: FontWeight.w300
              ),
            ),
          ]
        ),
      ),
      ListTile(
        leading: Icon(Icons.train),
        title: Text(
          "Tickets",
          style: Theme.of(context).textTheme.title,
        ),
      ),
      ListTile(title: Text("Buy Tickets"), onTap: () {}),
      ListTile(title: Text("My Tickets"), onTap: () {}),
      Divider(),
      ListTile(
        leading: Icon(Icons.person),
        title: Text(
          "Profile",
          style: Theme.of(context).textTheme.title,
        ),
      ),
      ListTile(title: Text("Profile information"), onTap: () {}),
      ListTile(title: Text("Past trips"), onTap: () {}),
      ListTile(title: Text("Loyalty program points"), onTap: () {}),
    ],
  )
),
```

First of all, it's important to clarify that just adding a drawer to a Scaffold causes the framework to automatically add a button to the app bar to expand it, as you can see with an empty Scaffold body as shown in the screenshot on page 11.



The Drawer itself looks like this:



At this point, you might not be sure about how to actually use the drawer for navigation purposes, and that is because it's not done using Navigator.push(), but by changing the state of the home page and some data that can be used by build() to build the page and, after doing that, calling Navigator.pop() to go back to the home page.

This means the home page has to be a stateful widget (don't forget that, especially if you get a *setState isn't defined for HomePage* error) and it is a great chance to introduce Dart enums.

Enumerated Types

You can create an enumerated type of data by code like the following:

```
enum PageType {
  buyTickets,
  myTickets,
  profileInfo,
  pastTrips,
  myPoints
}
```

which creates a new type called PageType which can only be one of PageType.buyTickets, PageType.myTickets, PageType.profileInfo, PageType.pastTrips, or PageInfo.myPoints.

Even if you are not used to this kind of data type, you'll surely be able to understand it with the continuation of the travel app example.

Finishing Our Travel App's Navigation

To finish the travel app's navigation, add the code on page 12 to your app outside any class definition and a complete, working example of a State that implements the functionality I described is the following:

```
class HomePageState extends State<HomePage> {

  PageType _pageType = PageType.buyTickets;
  String _str;
  String _sub;

  @override
  Widget build(BuildContext context) {
    switch(_pageType) {
      case PageType.buyTickets:
        _str = "Buy Tickets";
        _sub = "You can buy your tickets here";
        break;
      case PageType.myPoints:
        _str = "My Points";
        _sub = "You can buy check your loyalty program points here";
        break;
```

```
    case PageType.myTickets:
      _str = "My Tickets";
      _sub = "You can see the tickets you've bought here";
      break;
    case PageType.pastTrips:
      _str = "My Past Trips";
      _sub = "You can check the trips you have made previously here";
      break;
    case PageType.profileInfo:
      _str = "My Profile Information";
      _sub = "You can see your profile information here";
  }
  return Scaffold(
    appBar: AppBar(
      title: Text("Programming Travels"),
      backgroundColor: Colors.red,
    ),
    drawer: Drawer(
      child: ListView(
        padding: EdgeInsets.zero,
        children: [
          DrawerHeader(
            decoration: BoxDecoration(
              color: Colors.red,
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisAlignment: MainAxisAlignment.end,
              children: [
                Text(
                  "Profile Name",
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 18,
                  ),
                ),
                Text(
                  "email@example.com",
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: 11,
                    fontWeight: FontWeight.w300
                  ),
                ),
              ]
            ),
          ),
          ListTile(
            leading: Icon(Icons.train),
```

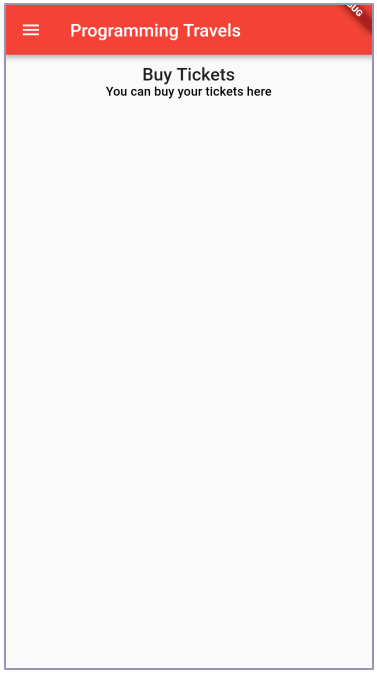
```
      title: Text(
        "Tickets",
        style: Theme.of(context).textTheme.title,
      ),
    ),
    ListTile(
      title: Text("Buy Tickets"),
      onTap: () {
        setState(
          () => _pageType = PageType.buyTickets,
        );
        Navigator.pop(context);
      }
    ),
    ListTile(
      title: Text("My Tickets"),
      onTap: () {
        setState(
          () => _pageType = PageType.myTickets,
        );
        Navigator.pop(context);
      }
    ),
    Divider(),
    ListTile(
      leading: Icon(Icons.person),
      title: Text(
        "Profile",
        style: Theme.of(context).textTheme.title,
      ),
    ),
    ListTile(
      title: Text("Profile Information"),
      onTap: () {
        setState(
          () => _pageType = PageType.profileInfo,
        );
        Navigator.pop(context);
      }
    ),
    ListTile(
      title: Text("Past Trips"),
      onTap: () {
        setState(
          () => _pageType = PageType.pastTrips,
        );
        Navigator.pop(context);
      }
    ),
```

```
            ListTile(
              title: Text("Loyalty Program Points"),
              onTap: () {
                setState(
                  () => _pageType = PageType.myPoints,
                );
                Navigator.pop(context);
              }
            ),
          ],
        )
      ),
      Center(
        child: Padding(
          padding: EdgeInsets.all(10.0),
          child: Column(
            children: [
              Text(_str, style: Theme.of(context).textTheme.title),
              Text(_sub, style: Theme.of(context).textTheme.subtitle)
            ],
          ),
        ),
      ),
    );
  }
}
```

As you can see, each of the Drawer's ListTile sets the _pageType variable (which, by default, is PageType.buyTickets), reloads the home page using setState() and then calls Navigator.pop() to close the drawer.

build() checks which of them is the current _pageType and sets two strings (a title and a subtitle) accordingly. Those two strings are then displayed in a centered and padded Column as shown in the screenshot on page 16.

The result is working navigation using a Drawer menu.


Programming Travels
Buy Tickets
You can buy your tickets here