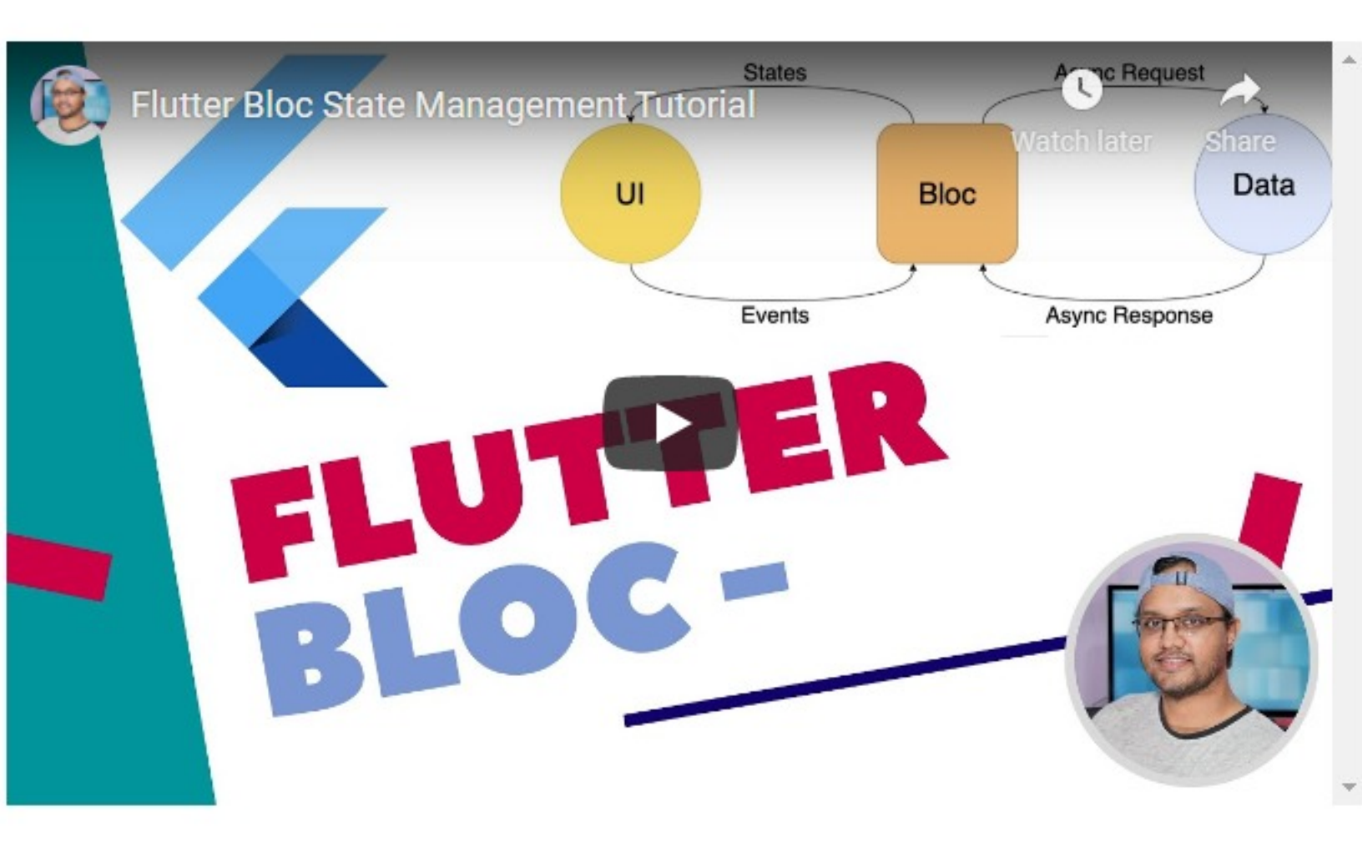
FLUTTER

How to Use Bloc in Flutter to Manage State

Learn to manage global state in a Flutter application

Mahmud Ahsan Follow
Aug 26, 2019 · 2 min read ★

Bloc is another state management library for flutter applications. B.Lo.C stands for **Business Logic Component**. Which takes 'Event' as input and produces 'State' as output. Bloc is built on top of RxDart. To understand some concepts, how to install, and how to use it in an application please check the video tutorial.



Source Code on GitHub

We can divide our flutter application architecture into 3 layers:

1. Presentation or View Layer
2. Bloc
3. Data Layer

So basically Bloc stands between View and Data layers.

1. Data Layer

Its responsibility is to provide data from any source. **Data Provider** provides raw data and the **repository** is a wrapper of one or more data providers.

2. Bloc or Business Logic

Its responsibility is to receive events from the presentation layer and respond with a new state. It works as a bridge between the data layer and the presentation layer.

3. Presentation Layer

Its responsibility is to render itself based on one or more bloc states. It also handles user input and application lifecycle events.

Some Concepts for Bloc

1. Events

Events are input to a Bloc. It's a similar concept of like `action` in Redux. In the presentation, layer events are generated by user interaction like a button click and passed to Bloc. An event may contain some additional data.

2. States

The state is part of the application's state. It is the output of a Bloc. When a state changed, UI components will get the notification, and based on the current state it can re-render itself.

3. Transition

The change from one state to another is called transition. It holds the current state, event, and the next state.

In Short:

- A `Bloc` must extend the base `Bloc` class from the core `bloc` package.
- Each `Bloc` must define an initial state
- It must implement a function called `mapEventToState`. This function takes incoming `event` as the argument. It also returns a `stream` of new `state` as output.
- To access the current bloc state we can use `currentState` property
- A bloc has a `dispatch` method. `Dispatch` takes an `event` and triggers `mapEventToState`. `Dispatch` can be called in the presentation layer or from within a bloc
- `onTransition` is called before a bloc state is updated
- `onError` can override to know bloc exception.

Here is the part of Bloc codes in the demo:

Step 1: Create Events

```
part of 'settings_bloc.dart';

@immutable
abstract class SettingsEvent {
  final dynamic payload;
  SettingsEvent(this.payload);
}

class FontSize extends SettingsEvent {
  FontSize(double payload) : super(payload);
}

class Bold extends SettingsEvent {
  Bold(bool payload) : super(payload);
}

class Italic extends SettingsEvent {
  Italic(bool payload) : super(payload);
}
```

flutter_bloc_1.dart hosted with ❤ by GitHub view raw

Step 2: Create a State

```
part of 'settings_bloc.dart';

@immutable
abstract class SettingsState {
  final double sliderFontSize;
  final bool isBold;
  final bool isItalic;
SettingsState({this.sliderFontSize, this.isBold, this.isItalic});
double get fontSize => sliderFontSize * 30;
}

class InitialSettingsState extends SettingsState {
  InitialSettingsState()
      : super(sliderFontSize: 0.5, isBold: false, isItalic: false);
}

class NewSettingState extends SettingsState {
  NewSettingState.fromOldSettingState(SettingsState oldState,
      {double sliderFontSize, bool isBold, bool isItalic})
      : super(
          sliderFontSize: sliderFontSize ?? oldState.sliderFontSize,
          isBold: isBold ?? oldState.isBold,
          isItalic: isItalic ?? oldState.isItalic,
        );
}
```

flutter_bloc_2.dart hosted with ❤ by GitHub view raw

Step 3: Create Bloc

```
import 'dart:async';
import 'package:bloc/bloc.dart';
import 'package:meta/meta.dart';

part 'settings_event.dart';
part 'settings_state.dart';

class SettingsBloc extends Bloc<SettingsEvent, SettingsState> {
  @override
  SettingsState get initialState => InitialSettingsState();

  @override
  Stream<SettingsState> mapEventToState(SettingsEvent event) async* {
    if (event is FontSize) {
      yield NewSettingState.fromOldSettingState(currentState,
          sliderFontSize: event.payload);
    } else if (event is Bold) {
      yield NewSettingState.fromOldSettingState(currentState,
          isBold: event.payload);
    } else if (event is Italic) {
      yield NewSettingState.fromOldSettingState(currentState,
          isItalic: event.payload);
    }
  }
}
```

flutter_bloc_3.dart hosted with ❤ by GitHub view raw

Step 4: Create BlocProvider in `main.dart`

```
import 'package:flutter/material.dart';
import 'package:states_bloc/home.dart';
import 'package:states_bloc/about.dart';
import 'package:states_bloc/settings.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:states_bloc/bloc/settings/settings_bloc.dart';

void main() {
  final BlocProvider<SettingsBloc> blocProvider = BlocProvider<SettingsBloc>(
    builder: (_) => SettingsBloc(),
    child: MyApp(),
  );
  runApp(blocProvider);
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => Home(),
        '/about': (context) => About(),
        '/settings': (context) => Settings(),
      },
    );
  }
}
```

flutter_bloc_4.dart hosted with ❤ by GitHub view raw

Step 5: Use Bloc state and dispatch events

```
import 'package:flutter/material.dart';
import 'package:states_bloc/drawer_menu.dart';
import 'package:states_bloc/bloc/settings/settings_bloc.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

class Settings extends StatelessWidget {
  double _value = 0.5;
  bool isBold = false;
  bool isItalic = false;
  @override
  Widget build(BuildContext context) {
    final SettingsBloc settingsBloc = BlocProvider.of<SettingsBloc>(context);
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.teal,
        title: Text('Settings'),
      ),
      drawer: DrawerMenu(),
      body: BlocBuilder<SettingsBloc, SettingsState>(
        builder: (context, state) {
          return Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: <Widget>[
              Padding(
                padding: EdgeInsets.only(left: 20, top: 20),
                child: Text(
                  'Font Size: ${state.fontSize.toInt()}',
                  style: TextStyle(
                      fontSize: Theme.of(context).textTheme.headline.fontSize),
                ),
              ),
              Slider(
                  min: 0.5,
                  value: state.sliderFontSize,
                  onChanged: (newValue) {
                    settingsBloc.dispatch(FontSize(newValue));
                  }),
              Container(
                margin: EdgeInsets.symmetric(horizontal: 8),
                child: Row(
                  children: <Widget>[
                    Checkbox(
                      value: state.isBold,
                      onChanged: (newVal) {
                        settingsBloc.dispatch(Bold(newVal));
                      },
                    ),
                    Text(
                      'Bold',
                      style: getStyle(state.isBold, false),
                    ),
                  ],
                ),
              ),
              Container(
                margin: EdgeInsets.symmetric(horizontal: 8),
                child: Row(
                  children: <Widget>[
                    Checkbox(
                        value: state.isItalic,
                        onChanged: (newVal) {
                          settingsBloc.dispatch(Italic(newVal));
                        }),
                    Text(
                      'Italic',
                      style: getStyle(false, state.isItalic),
                    ),
                  ],
                ),
              ),
            ],
          );
        },
      ),
    );
  }

TextStyle getStyle([bool isBold = false, bool isItalic = false]) {
    return TextStyle(
      fontSize: 18,
      fontWeight: isBold ? FontWeight.bold : FontWeight.normal,
      fontStyle: isItalic ? FontStyle.italic : FontStyle.normal,
    );
  }
}
```

flutter_bloc_5.dart hosted with ❤ by GitHub view raw

Dart Flutter Programming Mobile App Development Android App Development

170 claps 4 responses

WRITTEN BY
Mahmud Ahsan Follow
computer programmer

Level Up Programming Follow
Enhance skills, deliver more

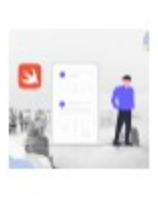
More From Medium

SwiftUI—How to Control ScrollView Programmatically
Mahmud Ahsan in Level Up Programming

SwiftUI—How to Create a Custom ActionSheet Card View
Mahmud Ahsan in Level Up Programming

iOS—How to Take Advantages of Multiple Schemes in XCode
Mahmud Ahsan in Level Up Programming

Swift—How to Use Delegate Effectively in iOS App Development
Mahmud Ahsan in Level Up Programming



How to Use Fastlane to Deploy iOS App Fast
Mahmud Ahsan in Level Up Programming

How to Reverse Engineering an Android App
Mahmud Ahsan in Level Up Programming

How to Use Provider in Flutter
Mahmud Ahsan in Level Up Programming

Dart Language Tutorial for Programmers
Mahmud Ahsan in Level Up Programming