BLoc Concepts

1. Using Cubit first
2. Import bloc

  flutter\_bloc: ^8.1.6

1. Crate first cubit we try it first

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class CounterCubit extends Cubit<int> {

  //must create a counter cubit constructor with initial state(here counter inilaly 0)

  CounterCubit() : super(0);

  //method to increment a counter

  void increment() {

    debugPrint("decrementing....");

    /\*

Updates the [state] to the provided [state]. [emit] does nothing if the [state] being emitted is equal to the current [state].

To allow for the possibility of notifying listeners of the initial state, emitting a state which is equal to the initial state is allowed as long as it is the first thing emitted by the instance.

Throws a [StateError] if the bloc is closed.

\*/

    emit(state + 1);

  }

  //method to decrement a counter

  void decrement() {

    debugPrint("decrementing....");

    //task 1 create a method to dcrement counter

    // emit(state - 1);

    //task 2 if counter is 0 stop decrementing

    //a)

    // emit(state < 1 ? state : state - 1);

    //b)

    if (state == 0) {

      return;

    }

    emit(state - 1);

  }

}

1. Create a home screen

import 'package:bloc\_pro/cubit/counter\_cubit.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class HomeUsingCubit extends StatefulWidget {

  const HomeUsingCubit({super.key});

  @override

  State<HomeUsingCubit> createState() => \_HomeUsingCubitState();

}

class \_HomeUsingCubitState extends State<HomeUsingCubit> {

  //creating instance of the class

  final counterCubit = CounterCubit();

  @override

  Widget build(BuildContext context) {

    //using a instance every time when UI re-builds (it will work with setState and this counter value).

    // final counter = counterCubit.state;

    debugPrint("build method is called");

    return Scaffold(

      appBar: AppBar(

        title: const Text("counter using cubit"),

      ),

      body: Center(

        child: BlocBuilder<CounterCubit, int>(

            //if we dont add block error=>Could not find the correct Provider<CounterCubit> above this BlocBuilder<CounterCubit, int> Widget

            bloc: counterCubit,

            builder: (context, value) {

              return Text(

                value.toString(),

                style: const TextStyle(

                  fontSize: 18,

                  fontWeight: FontWeight.bold,

                ),

              );

            }),

      ),

      floatingActionButton: Column(

        mainAxisAlignment: MainAxisAlignment.end,

        children: [

          FloatingActionButton(

            onPressed: () {

              counterCubit.increment();

            },

            child: const Icon(Icons.add),

          ),

          const SizedBox(

            height: 30,

          ),

          FloatingActionButton(

            onPressed: () {

              counterCubit.decrement();

            },

            child: const Icon(Icons.remove),

          ),

        ],

      ),

    );

  }

}

1. Task is to create separate files for those increment/decrement operations

We will user provider but how we are using block in program

Here we go we have provider in block only which name **BlockProvider**() having same functionality

**Syntax:**

**Bloc provider**

 final counterCubit = BlocProvider.of<CounterCubit>(context);

**To use it:**

**Here in bloc builder we passed bloc:** **counterCubit but if we using bloc provider in main.dart then it is not manditary**

BlocBuilder<CounterCubit, int>(

          //if we dont add block error=>Could not find the correct Provider<CounterCubit> above this BlocBuilder<CounterCubit, int> Widget

          bloc: counterCubit,

          builder: (context, value) {

            return Text(

              value.toString(),

              style: const TextStyle(

                fontSize: 18,

                fontWeight: FontWeight.bold,

              ),

            );

          },

        ),

1. Need to register in main.dart (same as provider)

import 'package:bloc\_pro/cubit/counter\_cubit.dart';

import 'package:bloc\_pro/cubit/home\_using\_cubit.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    /\*

    Takes a [Create] function that is responsible for creating the [Bloc] or [Cubit]

    and a [child] which will have access to the instance via BlocProvider.of(context).

    It is used as a dependency injection (DI) widget so that a single instance of

    a [Bloc] or [Cubit] can be provided to multiple widgets within a subtree.

     \*/

    return BlocProvider(

      create: (\_) => CounterCubit(),

      child: MaterialApp(

        title: 'Flutter Demo',

        theme: ThemeData(

          colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

          useMaterial3: true,

        ),

        // home: const MyHomePage(title: 'Flutter Demo Home Page'),

        home: const HomeUsingCubit(),

      ),

    );

  }

}

1. Now create different file for increment decrement (state less widget)

import 'package:bloc\_pro/cubit/counter\_cubit.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class UsingDiffFile extends StatelessWidget {

  const UsingDiffFile({super.key});

  @override

  Widget build(BuildContext context) {

    final counterCubit = BlocProvider.of<CounterCubit>(context);

    return Scaffold(

      appBar: AppBar(

        title: const Text("passing cubic"),

      ),

      floatingActionButton: Column(

        mainAxisAlignment: MainAxisAlignment.end,

        children: [

          IconButton(

            onPressed: () => counterCubit.increment(),

            icon: const Icon(Icons.add),

            color: Colors.deepPurple,

          ),

          IconButton(

            color: Colors.deepPurple,

            onPressed: () => counterCubit.decrement(),

            icon: const Icon(Icons.remove),

          ),

        ],

      ),

    );

  }

}

1. Now make home screen as state less and only user **Navigator.of(context).push()** do not user **Navigator.push()**

import 'package:bloc\_pro/cubit/counter\_cubit.dart';

import 'package:bloc\_pro/cubit/using\_diff\_file.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class HomeUsingCubit extends StatelessWidget {

  const HomeUsingCubit({super.key});

  @override

  Widget build(BuildContext context) {

    final counterCubit = BlocProvider.of<CounterCubit>(context);

    debugPrint("build method is called");

    return Scaffold(

      appBar: AppBar(

        title: const Text("counter using cubit"),

      ),

      body: Center(

        child: BlocBuilder<CounterCubit, int>(

          //if we dont add block error=>Could not find the correct Provider<CounterCubit> above this BlocBuilder<CounterCubit, int> Widget

          bloc: counterCubit,

          builder: (context, value) {

            return Text(

              value.toString(),

              style: const TextStyle(

                fontSize: 18,

                fontWeight: FontWeight.bold,

              ),

            );

          },

        ),

      ),

      floatingActionButton: FloatingActionButton(

        child: const Icon(Icons.navigate\_next),

        onPressed: () {

          Navigator.of(context).push(

            MaterialPageRoute(

              builder: (context) => const UsingDiffFile(),

            ),

          );

        },

      ),

    );

  }

}

1. Using Bloc for Counter
2. Create a class which extends bloc

import 'package:flutter\_bloc/flutter\_bloc.dart';

//increment event

class CounterIncrement {}

//bloc=>takes a Stream of Events as input and transforms them into a Stream of States as output.

/\*

abstract class Bloc<Event, State> extends BlocBase<State> implements BlocEventSink<Event>

\*/

class CounterBloc extends Bloc<CounterIncrement, int> {

  CounterBloc() : super(0) {

    //we can not add emit method directly in bloc so we need to create eveent handler

    /\*

void on<E extends CounterIncrement>(FutureOr<void> Function(E, Emitter<int>) handler, {Stream<E> Function(Stream<E>, Stream<E> Function(E))? transformer})

    \*/

    //on=>Register event handler for an event of type E. There should only ever be one event handler per event type E.

    on<CounterIncrement>(

      (event, emit) {

        emit(state + 1);

      },

    );

    /\*

    warning=>

    The member 'emit' can only be used within 'package:bloc/src/bloc.dart' or a test

    [emit] is only for internal use and should never be called directly outside of tests. The [Emitter] instance provided to each [EventHandler] should be used instead.

    \*/

    // emit(state + 1);

  }

}

1. Create a file on frontend

Here no need to create a instance of block provider block provides directly

import 'package:bloc\_pro/bloc/counter\_bloc.dart';

import 'package:bloc\_pro/bloc/diff\_file\_using\_block.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class HomeUsingBloc extends StatelessWidget {

  const HomeUsingBloc({super.key});

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("Home using bloc"),

      ),

      body: Center(

        child: BlocBuilder<CounterBloc, int>(

          builder: (context, counter) {

            return Text(

              counter.toString(),

              style: const TextStyle(fontWeight: FontWeight.bold, fontSize: 20),

            );

          },

        ),

      ),

      floatingActionButton: FloatingActionButton(

        onPressed: () {

          Navigator.of(context).push(

            MaterialPageRoute(

              builder: (context) => const DiffFileUsingBlock(),

            ),

          );

        },

        child: const Icon(Icons.navigate\_next),

      ),

    );

  }

}

1. Now create different file for method where we need to create instance of bloc provider

import 'package:bloc\_pro/bloc/counter\_bloc.dart';

import 'package:flutter/material.dart';

import 'package:flutter\_bloc/flutter\_bloc.dart';

class DiffFileUsingBlock extends StatelessWidget {

  const DiffFileUsingBlock({super.key});

  @override

  Widget build(BuildContext context) {

    final counterBlock = BlocProvider.of<CounterBloc>(context);

    return Scaffold(

      appBar: AppBar(

        title: const Text("bloc counter"),

      ),

      floatingActionButton: Column(

        mainAxisAlignment: MainAxisAlignment.end,

        children: [

          IconButton(

            onPressed: () {

              counterBlock.add(CounterIncrement());

            },

            icon: const Icon(Icons.add),

          ),

          IconButton(

            onPressed: () {},

            icon: const Icon(Icons.remove),

          ),

        ],

      ),

    );

  }

}

1. Now how to create a decrement event:

import 'package:flutter\_bloc/flutter\_bloc.dart';

//to pass multiple event throw bloc then we create a parent class and add other classes as their child(use sealed class as perent\*\*\*)

sealed class CounterEvent {}

//increment event

class CounterIncrement extends CounterEvent {}

//decrement event

final class CounterDecrement extends CounterEvent {}

class CounterBloc extends Bloc<CounterEvent, int> {

  CounterBloc() : super(0) {

    //increment

    on<CounterIncrement>(

      (event, emit) {

        emit(state + 1);

      },

    );

//decrement

    on<CounterDecrement>((event, emit) {

      emit(state - 1);

    });

  }

}

And do not forgot to bind this event to decrement sign in anther class

1. Now how to create a neat clean code

We will use part and part of without importing things like 🡺

Counter\_event.dart

part of 'counter\_bloc.dart';

//to pass multiple event throw bloc then we create a parent class and add other classes as their child(use sealed classes\*\*\*)

sealed class CounterEvent {}

//increment event

class CounterIncrement extends CounterEvent {}

//decrement event

final class CounterDecrement extends CounterEvent {}

//restart event

final class CounterRestart extends CounterEvent {}

counter\_bloc.dart

import 'package:flutter\_bloc/flutter\_bloc.dart';

//part

part 'counter\_event.dart';

class CounterBloc extends Bloc<CounterEvent, int> {

  CounterBloc() : super(0) {

    //increment

    on<CounterIncrement>(

      (event, emit) {

        emit(state + 1);

      },

    );

    //decrement

    on<CounterDecrement>((event, emit) {

      if (state == 0) {

        return;

      }

      emit(state - 1);

    });

    //restart

    on((event, emit) {

      emit(state - 0);

    });

  }

}