Trip and Travel App

Abstract

In this app, ı did state management with Cubit. There are two classes I created for them:

- AppCubits that extends cubit
- CubitStates that extends equatable

I used BlocProvider for create cubits in main function.

I build a flutter Cubit / Bloc state management app with ui and do api request.

I understood and did how to get http request and load the data using flutter cubit or bloc.

I use blocprovider and blocbuilder for state update.

This app covers in summary, building the model and API request.

Main Features

1. Custom Carousel Slider
2. TabBar and TabBarView & TivkerProviderState & Controllers
3. ListView.builder
4. BottomNavigationBar
5. Cubit State Management
6. Creating Cubit and States
7. BlocProvider and BlocBuilder
8. Cubit Navigation
9. Cubit Data Load From Server
10. Building Model
11. Data Repository.

Packages

- bloc
- flutter_bloc
- equatable
- http

What I did step by step (with class & widgets) :

Start new flutter project

Carousel Slider

Create a new dart class welcome_page.dart

Use pageview.builder for carousel

Create list of images for the slider

boxfit for images

Build column for the carousel

Build reusable widgets app_large_text

Build reusable widgets for app_text

Build reusable custom button with name “Responsive Button”

Build the dots for the carousel

Select the dots with the slider

Build main_page.dart and other nav pages

Build the bottomNavigationBar

BottomNavigationbarItem

Build home_page.dart

indexes for BottomNavigationbar

Style BottomNavigationBar

onTap Event for BottomNavigationbar

Work on home_page.dart

Finish building the icon

Tabbar and Tabbarview

TabBarController with TickerProviderStateMixin

Style Tabbar

Build Custom Indicator for TabBar, createBoxPainter and Paint

TabBar Indicator Position

TabBarView Build

Build the small images in row

Build the map of images and text

Access the images and text using map.keys.elementAt(index) and map.values.element(index)

Idea of the the detail_page.dart

Page architecture stack and positioned widget

Restructure the project folders

Build the detail_page.dart

Header image for detail page

Build mid section of the detail page

Build the column layout for the detail page

Star widgets

Build the description section

Working on resubale app large buttons responsive_buttons.dart

Implementing cubit state management

Build cubit folder, cubit class and state class

Working on states class app_cubit_states.dart

Create app_cubits.dart class

Create BlocProvider

Build child widget for BlocProvider app_cubit_logics.dart

Conditional check the states using BlocBuilder

Using the first flutter cubit

Installing http

Work on the data_services.dart and work on the rest API

Build the model data_model.dart

emit() state for Loading and Loaded state

NavigatetoanewpageusingBlocProvider.of

Create a state for theloadeddataandemitthestate

Gotohome_dart.fileusingBlocBuilder

Grab the Loaded data inside home_page.dart using BlocBuilder from state and replace the static data

Create a new state an demit() for detail_page.dart

Navigate to DetailPage using BlocProvider.of

Find the DetailPage() in app_cubit_logics.dart

Go back to the previous page

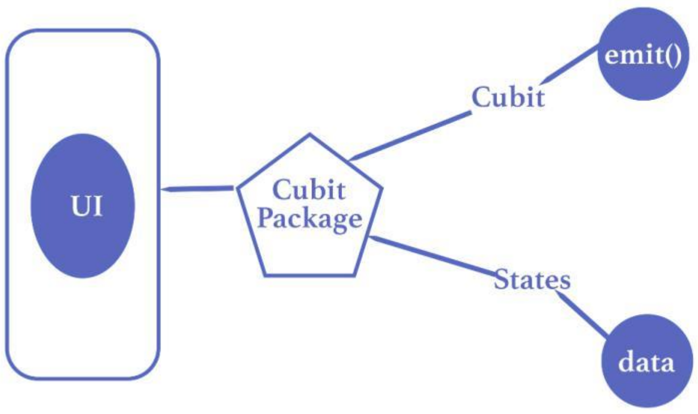
Update the data dynamically on detail_page.dart

Run the whole app

Fix th ebottom navigation part for the main_page.dart

Finish

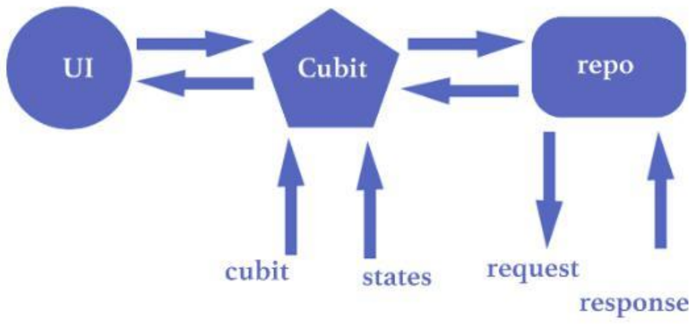
What is Cubit and How to Use Basicly



A Cubit is similar to Bloc but has no notion of events and relies on methods to emit new states. Simply speaking, cubit contains states and based on request it emits(fires) the states.

See another diagram for cubit with network request

Cubit paketini kurduktan sonra cubit ve state class'larımı oluşturdum.



Here we see that, we can accomplish network request using cubit.

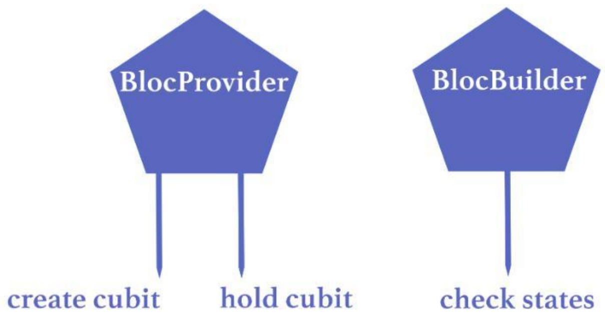
Two other important concepts about Cubit. It has two important classes. I used them through out the app:

1) BlocProvider

This one creates cubit for the app, and also specify the entry point for using cubit

2) BlocBuilder

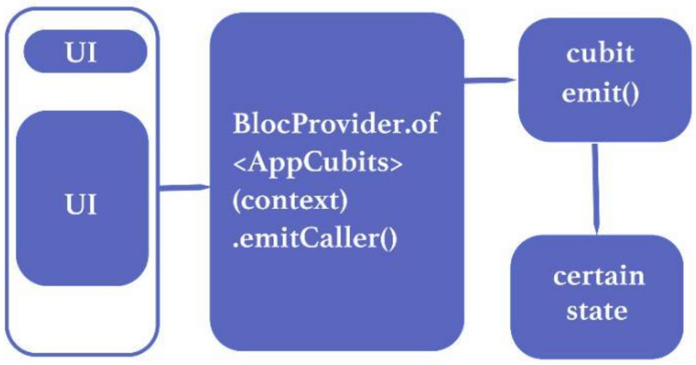
This one checks all the states and ask for ui change



Every Cubit requires an initial state which will be the state of the Cubit before emit has been called.

Just like BLoc we need to create a Cubit first. We will create a Cubit using BlocProvider just like Bloc creating.

Accessing cubit and state from ui



At the core, you need to use BlocProvider.of<AppCubits>(context).emitCaller() to access the cubit(the certain emit() function and the state. See the flow how it works.