**Thiết kế giao diện Fluter**

* [Flutter là gì?](https://salesoft.vn/tin-tuc/flutter-la-gi-gioi-thieu-ve-flutter#flutter-l%C3%A0-g%C3%AC?)
* [Lịch sử hình thành Flutter?](https://salesoft.vn/tin-tuc/flutter-la-gi-gioi-thieu-ve-flutter#l%E1%BB%8Bch-s%E1%BB%AD-h%C3%ACnh-th%C3%A0nh-flutter?)
* [Tại sao nên sử dụng Flutter ?](https://salesoft.vn/tin-tuc/flutter-la-gi-gioi-thieu-ve-flutter#t%E1%BA%A1i-sao-n%C3%AAn-s%E1%BB%AD-d%E1%BB%A5ng-flutter-?)

 Với sự gia tăng đáng kinh ngạc của sự phát triển ứng dụng di động trong thập kỷ qua, iOS và Android đã trở thành những hệ điều hành hàng đầu. Điều này dẫn đến sự trỗi dậy của toàn bộ ngành công nghiệp với một số framework cho phép các nhà phát triển xây ứng dụng di động. Flutter là thành viên mới nhất trong số các framework phát triển di động. Hãy cùng VTS đi tìm hiểu nó bằng bài viết dưới đây nhé!

Flutter là gì?



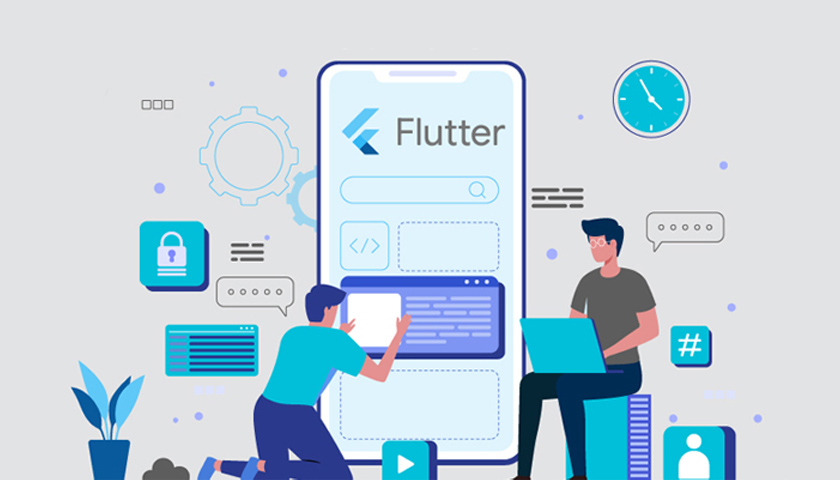
*Flutter là gì?*

[***Flutter***](https://salesoft.vn/tin-tuc/flutter-la-gi-gioi-thieu-ve-flutter) là bộ công cụ phát triển triển giao diện phần mềm người dùng được Google phát triển, Flutter được sử dụng để phát triển các [ứng dụng di động](https://salesoft.vn/mobile-app-development) đa nền tảng như iOS, Android, MacOS, Linux, Google Fuchisia và web trên một ngôn ngữ lập trình duy nhất.

Flutter gồm 2 thành phần quan trọng:

* ***Một SDK (Software Development Kit)***: Một bộ sưu tập các công cụ sẽ giúp bạn phát triển các ứng dụng của mình. Điều này bao gồm các công cụ để biên dịch mã của bạn thành mã máy gốc (mã cho iOS và Android).
* ***Một Framework (UI Library based on widgets)***: Một tập hợp các thành phần giao diện người dùng (UI) có thể tái sử dụng (button, text inputs, slider, v.v.) giúp bạn có thể cá nhân hóa tùy theo nhu cầu của riêng mình.

Lịch sử hình thành Flutter?



*Lịch sử hình thành Flutter*

***Flutter*** được ra mắt lần đầu năm 2015 và chính thức phát hành vào tháng 5/2017. Cho đến nay, Flutter được biết đến với khá nhiều các phiên bản:

* *Phiên bản đầu tiên của Flutter được biết đến với tên gọi "SKY" và chạy trên hệ điều hành Adroid. Fultter được ra mắt lần đầu tại hội nghị thượng đỉnh các nhà phát triển Dart năm 2015.*
* *Tháng 5/2020 bộ công cụ phát triển phần mềm Dart (SDK) phiên bản 2.8 và Flutter 1.17/0 đã được phát hành bổ sung thêm bộ hỗ trợ Metal API giúp cải thiện 50% hiệu suất trên thiết bị IOS.*
* *Phiên bản Flutter 2.0 được phát hành vào tháng 2/2021 trong một sự kiện trực tuyến Flutter Engage. Phiên bản mới này chính thức hỗ trợ phát triển ứng dụng trên Web, Windows, MACOS và Linux. Flutter 2.0 sử dụng Dart 2.0 giúp ứng dụng an toàn hơn.*
* *Phiên bản Flutter 2.5 và Dart 2.14 được phát hành bởi Google vào tháng 9/2021, phiên bản này đã cải tiến chế độ toàn màn hình của Adroid và phiên bản Material Design mới nhất của Google có tên là Material You.*
* *Phiên bản Flutter 3 và Dart 2.17 được Google công bố vào Tháng 5/2022. Phiên bản này đã nâng cấp hỗ trợ 6 nền tảng bao gồm hỗ trợ ổn định cho Linux và macOS trên cả bộ vi xử lý Intel và Apple Silicon.*

Tại sao nên sử dụng Flutter ?



*Tại sao nên sử dụng Flutter?*

Mỗi công nghệ mới đều có những lợi thế riêng giúp chúng ta bằng cách này hay cách khác. Chúng ta hãy xem xét một số ưu điểm của Flutter để biến nó thành SDK thân thiện với nhà phát triển.

* ***Flutter giúp phát triển ứng dụng nhanh chóng***: Tính năng "hot reload" của nó giúp bạn nhanh chóng và dễ dàng thử nghiệm, xây dựng giao diện người dùng, thêm tính năng và sửa lỗi nhanh hơn. Trải nghiệm tải lại lần thứ hai, mà không làm mất trạng thái, trên emulator, simulator và device cho iOS và Android.
* ***Bộ UI đẹp và biểu cảm***: Thỏa mãn người dùng của bạn với các widget built-in đẹp mắt theo Material Design và Cupertino (iOS-flavor), các API chuyển động phong phú, scroll tự nhiên mượt mà và tự nhận thức được nền tảng.
* ***SDK native***: làm cho ứng dụng trở lên sống động với API của platform, SDK của bên thứ ba và native code, bạn cũng có thể sử dụng lại các mã Jave, Swift, Objc của bạn và truy cập các tính năng SDK native trên iOS và Android.
* ***Phát triển ứng dụng thống nhất:*** Flutter có các công cụ và thư viện để giúp bạn dễ dàng đưa ý tưởng của mình vào cuộc sống trên iOS và Android. Nếu bạn chưa có kinh nghiệm phát triển trên thiết bị di động, thì Flutter là một cách dễ dàng và nhanh chóng để xây dựng các ứng dụng di động tuyệt đẹp. Nếu bạn là một nhà phát triển iOS hoặc Android có kinh nghiệm, bạn có thể sử dụng Flutter cho các View của bạn và tận dụng nhiều code Java / Kotlin / ObjC / Swift hiện có của bạn.
* ***Framework hiện đại:*** Dễ dàng tạo giao diện người dùng của bạn với framework hiện đại của Flutter và tập hợp các platform, layout và widget phong phú. Giải quyết các thách thức giao diện người dùng khó khăn của bạn với các API mạnh mẽ và linh hoạt cho 2D, animation, gesture, hiệu ứng và hơn thế nữa.

**Bài 2. Hello Word**

Ứng dụng được viết trên ***Adroid Studio*** sử dụng ***Flutter*** và ngôn ngữ ***Dart***

Mọi thứ trong***Flutter*** đều là ***Widget, các Widget*** đã được định nghĩa sẵn và bạn chỉ cần sử dụng. Có 3 loại Widget trong Flutter

**-        Stateless Widget**

**-        Stateful Widget**

**-        Inherited Widget**

Trong ứng dụng này chúng tôi sẽ sử dụng ***Stateless Widget, Material App, Center và Text Widget***

***Stateless Widget*** trong Flutter là Widget không có phép thay đổi trạng thái của màn hình, nó có phương thức được gọi là build có nhiệm vụ hiển thị màn hình một lần duy nhất và không thay đổi dù bạn có thực hiện bất kỳ các function nào. Để vẽ lại màn hình thì bạn phải sử dụng **Statefull Widget**

***MaterialApp*** là một widget được cung cấp sẵn bởi ***Flutter Team, MaterialApp*** là một lớp chứa một loạt các thành phần, nó được ví như ngôi nhà mà trong đó chứa các widget khác

***Center Widget*** : Center cũng là một widget được cung cấp sẵn bởi Flutter Team, dùng để căn giữa màn hình và nó thường đi kèm với child

Center(

child: Widget(

),

),

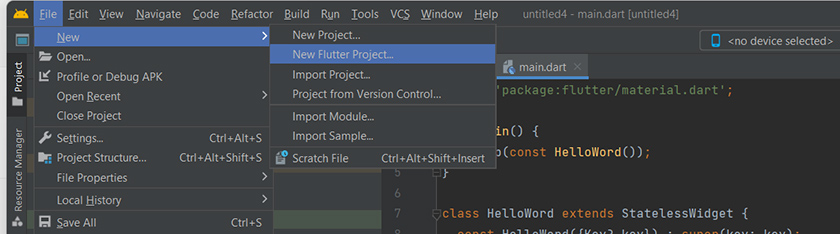
***Text Widget*** là một widget dùng để hiển thị văn bản trên màn hình

Sau khi đã hiểu một số widget được sử dụng trong bài viết, bây giờ các bạn hãy bắt tay vào viết ứng dụng Flutter đầu tiên nhé

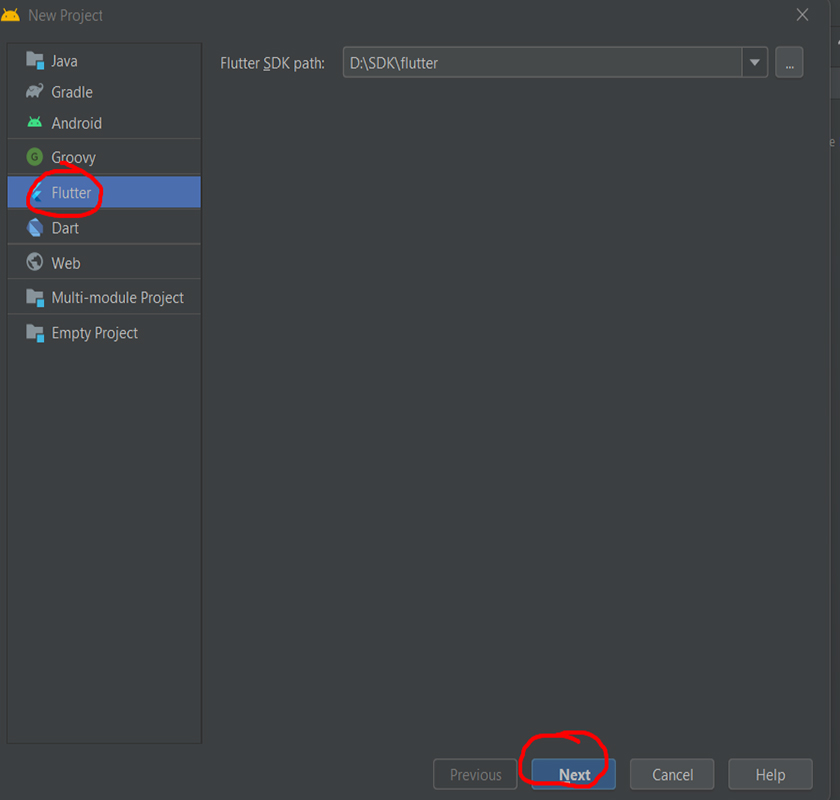
-        Tạo một ứng dụng mới

Để tạo ứng dụng mới trước tiên bạn phải khởi động Adroid Studio

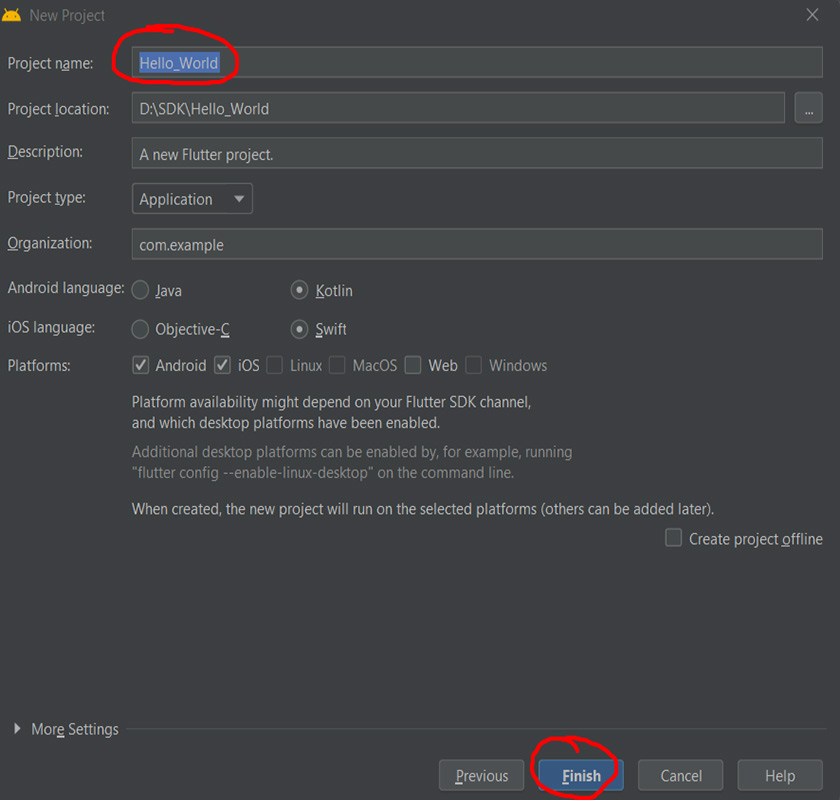
Vào menu File -> New -> New Fluter Project



Chọn Flutter -> Kích vào nút Next



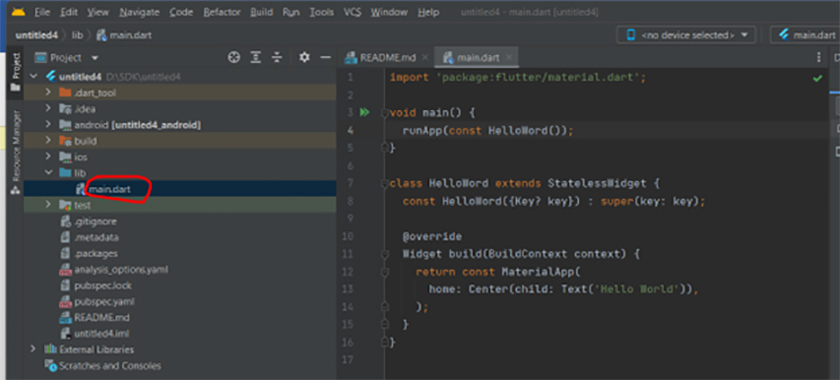
Nhập tên dự án vào ô Project name , trong VD này bạn đặt tên dự án là Hello\_World



Kích vào nút Finish để chúng ta hoàn thành bước tạo dự án mới

Viết đoạn lệnh cho ứng dụng

Mở file lib/main.dart



Xóa nội dung cũ trong file main.dart và sau đó copy code dưới đây vào thay thế

import 'package:flutter/material.dart';

void main() {

runApp(const HelloWord());

}

class HelloWord extends StatelessWidget {

const HelloWord({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

// Material App

return MaterialApp(

// Scaffold Widget

home: Scaffold(

appBar: AppBar(

// AppBar takes a Text Widget

// in it's title parameter

title: const Text('VTSOFT'),

),

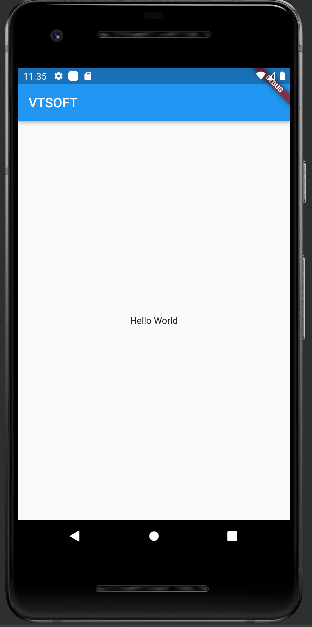
body: const Center(child: Text('Hello World')),

));

}

}

Chạy ứng dụng sẽ ra kết quả như sau :



Như vậy chúng ta đã hoàn thành viết APP đầu tiên, APP này còn khá đơn giải và chưa có các trang trí giao diện để cho đẹp hơn, trong các bài tiếp theo tôi sẽ hướng dẫn các bạn cách trang trí giao diện, bổ sung thêm các widget khác nhé

Giải thích quy trình thực hiện đoạn mã lệnh

* -        Đầu tiên các bạn khai báo sử dụng thư viện material cho toàn bộ APP

import 'package:flutter/material.dart';

* -        Tiếp theo khi chạy ứng dụng, hệ thống sẽ tìm đến hàm main để thực hiện, trong hàm main lại gọi đến hàm HelloWord()

runApp(const HelloWord());

Trong hàm main được kế thừa từ StatelessWidget, đây là một cấu trúc hình cây cơ bản của ứng dụng "hello World", khi chạy nó sẽ trả về một MaterialApp

Trước khi trả về một MeterialApp hệ thống sẽ thực hiện một Scaffold widget, Scaffold widget chứa hầu hết toàn bộ màn hình của ứng dụng, trong vd này Scaffold có chứa appBar dùng để hiển thị tiêu đề của APP "VTSOFT"

appBar: AppBar(

// AppBar takes a Text Widget

// in it's title parameter

title: const Text('VTSOFT'),

),

Phần body của ứng dụng có thuộc tính Center dùng để căn giữa, lớp Child nằm bên trong body lại chứ widget Text dùng để hiển thị văn bản

body: const Center(child: Text('Hello World')),

**Bài 3.  Tìm hiểu lớp Scaffold và các thành phần trong Flutter**

* [Scaffold là gì?](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#scaffold-l%C3%A0-g%C3%AC?)
* [Cấu trúc Scaffold](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#c%E1%BA%A5u-tr%C3%BAc-scaffold)
  + [FloatingActionButton](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#floatingactionbutton)
  + [AppBar](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#appbar)
  + [Drawer](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#drawer)
  + [BottomSheet](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#bottomsheet)
  + [BottomNavigationBar](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#bottomnavigationbar)
  + [SnackBar](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#snackbar)
  + [backgroundColor](https://salesoft.vn/tin-tuc/tim-hieu-lop-scaffold-va-cac-thanh-phan-trong-flutter#backgroundcolor)

Scaffold là gì?

Scaffold là một class trong ***Flutter, Scaffold***thông qua API cung cấp nhiều ***Widget*** để hiển thị như ***Drawer, SnackBar, BottomNavigationBar, FloatingActionButton, AppBar***, … ***Scaffold*** có khả năng mở rộng để lấp đầy màn hình hoặc không gian có sẵn.

Cấu trúc Scaffold

const Scaffold({

Key key,

PreferredSizeWidget appBar,

Widget body,

Widget floatingActionButton,

FloatingActionButtonLocation floatingActionButtonLocation,

FloatingActionButtonAnimator floatingActionButtonAnimator,

List persistentFooterButtons,

Widget drawer,

Widget endDrawer,

Widget bottomNavigationBar,

Widget bottomSheet,

Color backgroundColor,

bool resizeToAvoidBottomPadding,

bool resizeToAvoidBottomInset,

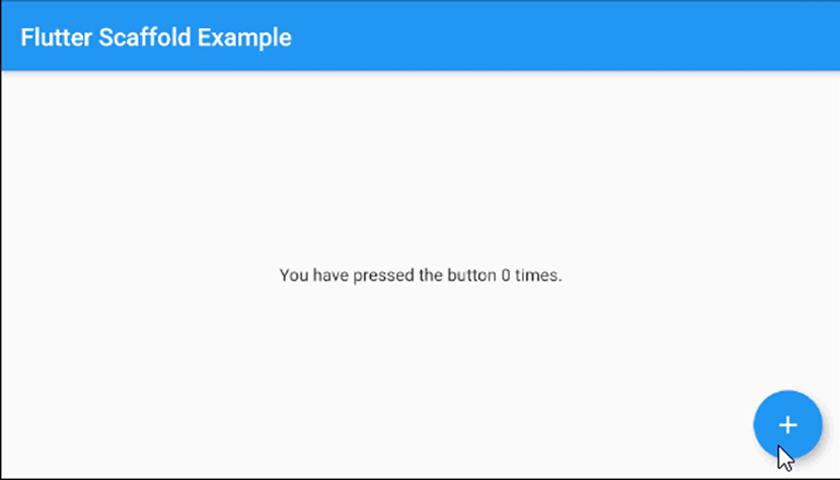
bool primary: true,

DragStartBehavior drawerDragStartBehavior: DragStartBehavior.down

})

1. **FloatingActionButton**

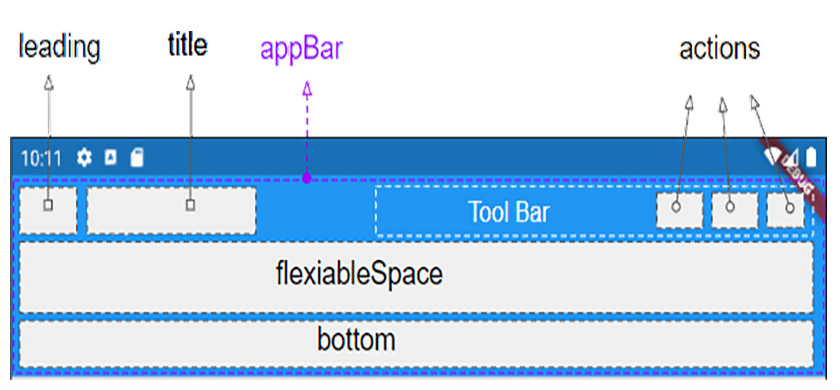
Là một biểu tượng hình tròn nổi phía trên màn hình, khi tác động vào sẽ thực thi một nhiệm vụ trong ứng dụng



1. **AppBar**

***AppBar*** hay còn gọi là Thanh ứng dụng, bao gồm một thanh công cụ (Tool Bar) và các widget đặc biệt khác

***AppBar*** được chia làm 5 nhóm khu vực gồm ***Leading, title, Tool Bar, flexibleSpace, bottom***

**

home: Scaffold(

appBar: AppBar(

// AppBar takes a Text Widget

// in it's title parameter

title: const Text('VTSOFT'),

),

body: const Center(child: Text('Hello World')),

));

1. **Drawer**

***Drawer*** là một Panel thường được hiển thị bên trái của màn hình, thông thường nó ở trạng thái ẩn và chỉ xuất hiện khi người sử dụng kích vào một chức năng trên ***AppBar*** hoặc vuốt từ trái qua phải trên màn hình.

drawer: Drawer(

child: ListView(

children: const [

DrawerHeader(

decoration: BoxDecoration(

color: Colors.green,

),

child: Text(

'Hello World',

style: TextStyle(

color: Colors.green,

fontSize: 24,

),

),

),

ListTile(

title: Text('Gallery'),

),

ListTile(

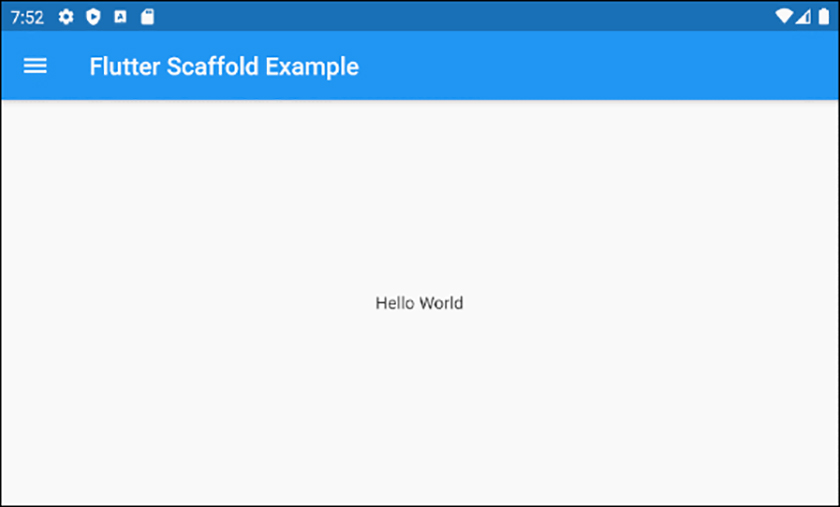
title: Text('Slideshow'),

),

],

)

),



1. **BottomSheet**

***BottomSheet*** là một khu vực thường được dùng để hiện thị các thông tin thêm (thông tin bổ sung) như thông tin liên hệ, nó được dùng để hiển thị phía gần cuối của Scaffold, không thay đổi vị trí khi người dùng cuộn màn hình lên hay xuống.

bottomSheet: Container(

height: 55,

color: Colors.cyan[50],

child:Column (

children: [

Row (

children: [

Icon(Icons.place),

SizedBox(width:5, height:5),

Text("Số 22/5 Lương Khánh Thiện, Hà Nội")

],

),

Row (

children: [

Icon(Icons.phone),

SizedBox(width:5, height:5),

Text("(098) 333-0380)")

],

)

],

)

),



1. **BottomNavigationBar**

**bottomNavigationBar** hay còn gọi là thanh điều hướng, nó được hiển thì phía cuối của **Scaffold** và cố định ngay cả khi bạn cuộn màn hình. Bạn có thể sử dụng **BottomAppBar** và **BottomNativationBar** trong thanh điều hướng này

bottomNavigationBar : BottomNavigationBar(

currentIndex : 0,

fixedColor : Colors.green,

items : [

BottomNavigationBarItem(

title : Text("Home"),

icon : Icon(Icons.home),

),

BottomNavigationBarItem(

title : Text("Search"),

icon : Icon(Icons.search),

),

BottomNavigationBarItem(

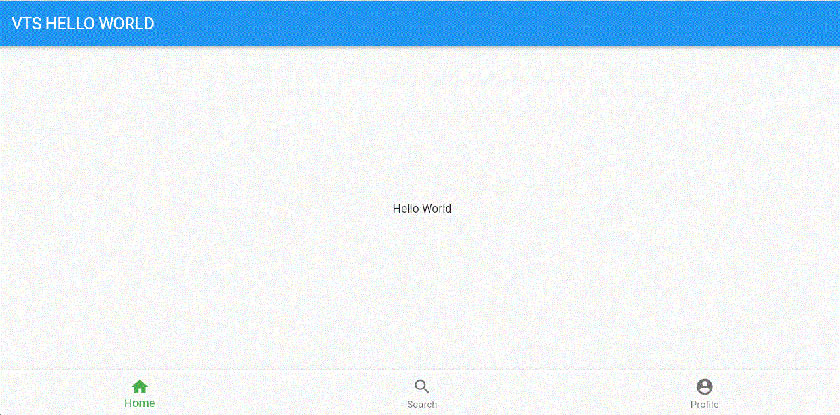
title : Text("Profile"),

icon : Icon(Icons.account\_circle),

),

],

)



1. **SnackBar**

**SnackBar** được sử dụng để hiển thị một thông báo đến người dùng và sẽ tự động được tắt đi sau khoảng một thời gian nhất định hoặc sau khi người dùng tương tác trên màn hình, nó được mặc định hiển thị phía cuối của **Scraffold**

Trong **SnackBar** bạn cũng có thể bổ sung thêm một hành động (action) để tương tác lại với thông báo như hành động UNDO để hủy bỏ kết quả đã thực hiện trước đó

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'o7planning.org',

debugShowCheckedModeBanner: false,

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Flutter SnackBar Example'),

),

body: Center(

child: Builder(

builder: (BuildContext ctxOfScaffold) {

return ElevatedButton(

child: Text('Show SnackBar'),

onPressed: () {

this.\_showSnackBarMsgDeleted(ctxOfScaffold);

}

);

}

)

)

);

}

void \_showSnackBarMsgDeleted(BuildContext ctxOfScaffold) {

// Create a SnackBar.

final snackBar = SnackBar(

content: Text('Message is deleted!'),

action: SnackBarAction(

label: 'UNDO',

onPressed: () {

this.\_showSnackBarMsgRestored(ctxOfScaffold);

},

),

);

// Find the Scaffold in the widget tree

// and use it to show a SnackBar.

Scaffold.of(ctxOfScaffold).showSnackBar(snackBar);

}

void \_showSnackBarMsgRestored(BuildContext ctxOfScaffold) {

// Create a SnackBar.

final snackBar = SnackBar(

content: Text('Message is restored!')

);

// Find the Scaffold in the widget tree

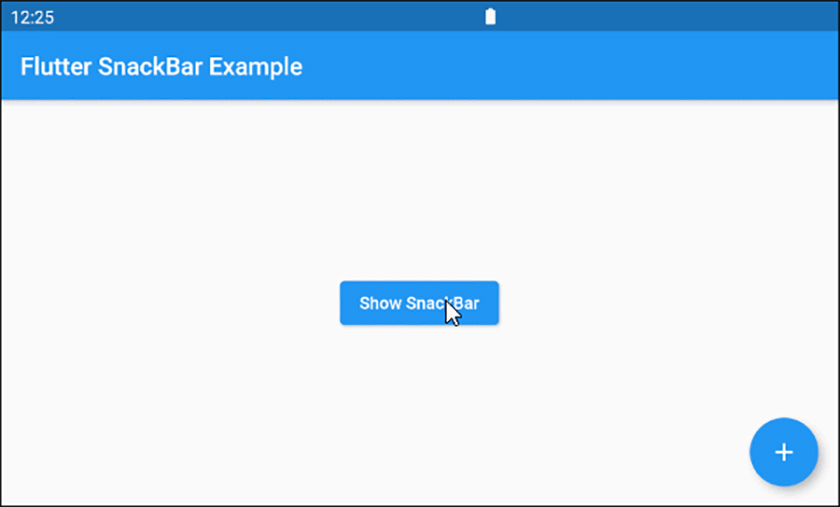
// and use it to show a SnackBar.

Scaffold.of(ctxOfScaffold).showSnackBar(snackBar);

}

}

Kết quả sau khi thực hiện



1. **backgroundColor**

***backgroundColor*** là một thuộc tính (property) được dùng để thiết lập màu nền cho toàn bộ màn hình bên trong Scraffold

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'o7planning.org',

debugShowCheckedModeBanner: false,

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

backgroundColor: Colors.greenAccent,

appBar: AppBar(

title: Text('Flutter SnackBar Example'),

),

body: Center(

child: Builder(

builder: (BuildContext ctxOfScaffold) {

return ElevatedButton(

child: Text('CÔNG TY CP TIN HỌC VÂN THANH'),

onPressed: () {

}

);

}

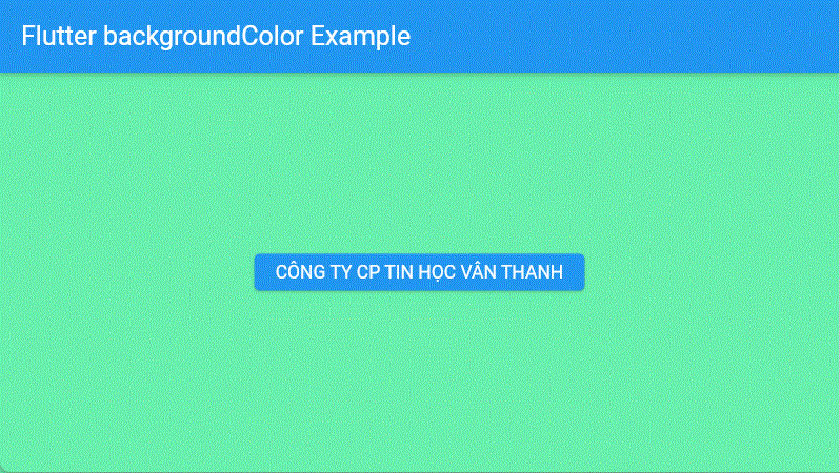
)

)

);

}

}



**Bài 3. AppBar trong Flutter**

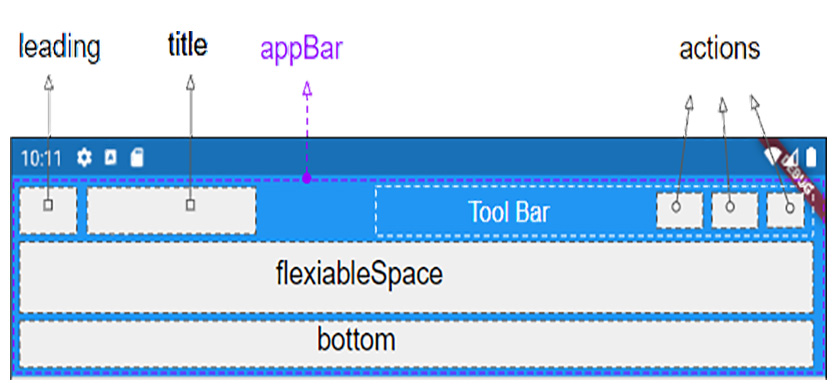
* [AppBar là gì?](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#appbar-l%C3%A0-g%C3%AC?)
  + [Cấu trúc của AppBar](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#c%E1%BA%A5u-tr%C3%BAc-c%E1%BB%A7a-appbar)
  + [Leading và cách sử dụng](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#leading-v%C3%A0-c%C3%A1ch-s%E1%BB%AD-d%E1%BB%A5ng)
  + [AutomaticallyImplyLeading](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#automaticallyimplyleading%C2%A0)
  + [Thuộc tính title](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#thu%E1%BB%99c-t%C3%ADnh-title)
  + [Căn lề title trong AppBar](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#c%C4%83n-l%E1%BB%81-title-trong-appbar)
  + [Chèn biểu tượng vào AppBar](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#ch%C3%A8n-bi%E1%BB%83u-t%C6%B0%E1%BB%A3ng-v%C3%A0o-appbar)
  + [FlexiableSpace](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#flexiablespace)
  + [Action](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#action)
  + [Bottom](https://salesoft.vn/tin-tuc/huong-dan-chi-tiet-cach-su-dung-appbar-trong-flutter#bottom)

AppBar là gì?

***AppBar*** hay còn gọi là Thanh công cụ thuộc lớp ***Scraffold***, nó dùng để miêu tả một chức năng hoặc một màn hình nào đó tới người dùng

***AppBar*** được đặt cố định phía trên cùng màn hình (hay trên cùng Scraffold), có chiều cao cố định

***AppBar*** là một lớp chứa bao gồm 5 khu vực *leading, title, Tool Bar (actions), flexiableSpace, bottom.*

**

Cấu trúc của AppBar

AppBar( {Key key,

Widget leading,

bool automaticallyImplyLeading: true,

Widget title,

List actions,

Widget flexibleSpace,

PreferredSizeWidget bottom,

double elevation,

Color shadowColor,

ShapeBorder shape,

Color backgroundColor,

Brightness brightness,

IconThemeData iconTheme,

IconThemeData actionsIconTheme,

TextTheme textTheme,

bool primary: true,

bool centerTitle,

bool excludeHeaderSemantics: false,

double titleSpacing: NavigationToolbar.kMiddleSpacing,

double toolbarOpacity: 1.0,

double bottomOpacity: 1.0,

double toolbarHeight

}

)

Leading và cách sử dụng

***Leading*** là một widget hiển thị ngay phía trước của title và nằm bên trái của màn hình

Kiểu của leading có thể là Icon hoặc IconButton

Ví dụ với kiểu là ***IconButton***

appBar: AppBar(

title: Text("VTS AppBar Title"),

leading: IconButton(

icon: Icon(Icons.notifications\_active),

onPressed: () {

// Do something.

}

)

)

Ví dụ về tạo một ***IconBotton*** trong khu vực leading khi kích vào IconBotton sẽ xuất hiện một thông báo

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: LeadingPage (),

);

}

}

class LeadingPage extends StatelessWidget {

const LeadingPage ({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS Leading Sample"),

leading: IconButton(

icon: Icon(Icons.notifications\_active),

onPressed: () {

showAlert(context);

}

)

),

body: Center(

child: Text("VTS chuyên đào tạo Flutter từ đơn giản đến chuyên sâu")

),

);

}

void showAlert(BuildContext context) {

showDialog(

context: context,

builder: (context) => AlertDialog(

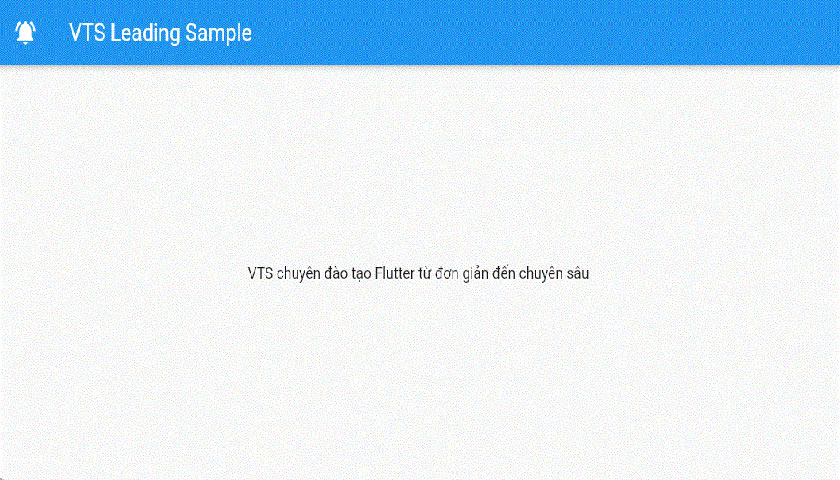
content: Text("Hi"),

));

}

}

Khi chạy ứng dụng sẽ ra kết quả sau



VD 2 : Hướng dẫn cách tạo ứng dụng có nút tắt/ bật âm thanh.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

const MyHomePage({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS Leading Title"),

leading: MyVolumeButton()

),

body: Center(

child: Text("VTS nhận đào tạo lập trình Flutter từ A-Z")

),

);

}

}

class MyVolumeButton extends StatefulWidget {

const MyVolumeButton({Key? key}) : super(key: key);

@override

State createState() {

return MyVolumeButtonState();

}

}

class MyVolumeButtonState extends State {

bool volumeOn = true;

@override

Widget build(BuildContext context) {

return IconButton(

icon: this.volumeOn? Icon(Icons.volume\_up):Icon(Icons.volume\_mute),

onPressed: () {

// Set new State

setState(() => this.volumeOn = !this.volumeOn);

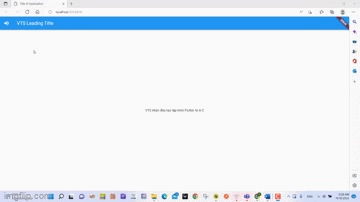
}

);

}

}

Chạy ứng dụng sẽ ra kết quả sau



AutomaticallyImplyLeading

***AutomaticallyImplyLeading***là một thuộc tính (property) có giá trị mặc định là True trong lớp AppBar. Nếu không đặt bất kỳ một Widget nào trong AppBar ứng dụng sẽ tự động thêm vào một Widget phù hợp theo từng ngữ cảnh

VD 1 : Khi bạn tạo một đối tượng Drawer (Ngăn xếp) trên AppBar một đối tượng IconButton sẽ được tự động thêm vào khu vực leading nếu

* *Vùng leading của AppBar không có đối tượng nào được khai báo (Rỗng)*
* *Thuộc tính AppBar. automaticallyImplyLeading : true*
* *AppBar được khai báo trong Scaffold*
* *Scaffold có chứa một Drawer*

VD 2 : Một IconButtom BACK sẽ tự động được thêm vào khu vực leading của AppBar để hỗ trợ người dùng quay trở lại màn hình trước nếu

* *Vùng leading của AppBar không có đối tượng nào được khai báo (Rỗng)*
* *Thuộc tính AppBar. automaticallyImplyLeading : true*
* *Không có AppBar.drawer nào được khai báo AppBar.drawer : null*
* *Bạn đang ở màn hình hiện tại từ việc di chuyển màn hình trước đó*

Ví dụ về automaticallyImplyLeading

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

// AppBar with automaticallyImplyLeading = "true" (Default)

appBar: AppBar(

title: Text("AppBar Title"),

automaticallyImplyLeading: true

)

body: Center(

child: Text("VTS tuyển sinh lập trình Flutter")

),

drawer: Drawer(

child: ListView

children: const [

DrawerHeader(

decoration: BoxDecoration(

color: Colors.green,

),

child: Text(

'My Drawer',

style: TextStyle(

color: Colors.green,

fontSize: 24,

),

),

),

ListTile(

title: Text('Gallery'),

),

ListTile(

title: Text('Slideshow'),

),

],

),

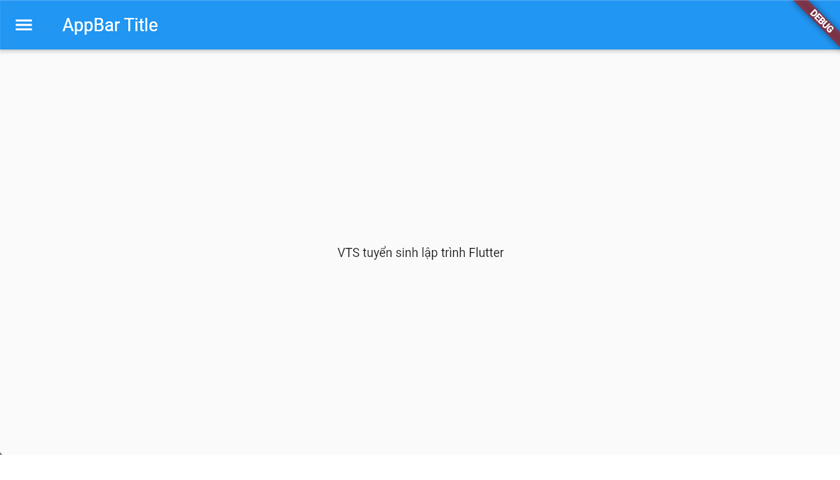
),

);

}

}

Chạy ứng dụng sẽ ra kế quả màn hình như sau



Thuộc tính title

Tilte (tiêu đề) dùng để hiển thị một đoạn văn bản trên Scraffold, nó thường dùng miên tả ngắn ngọn một chức năng hoặc nhiệm vụ nào đó

Ví dụ một AppBar đơn giản

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

const MyHomePage({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS AppBar Title"),

),

body: Center(

child: Text(

'Hello World',

)

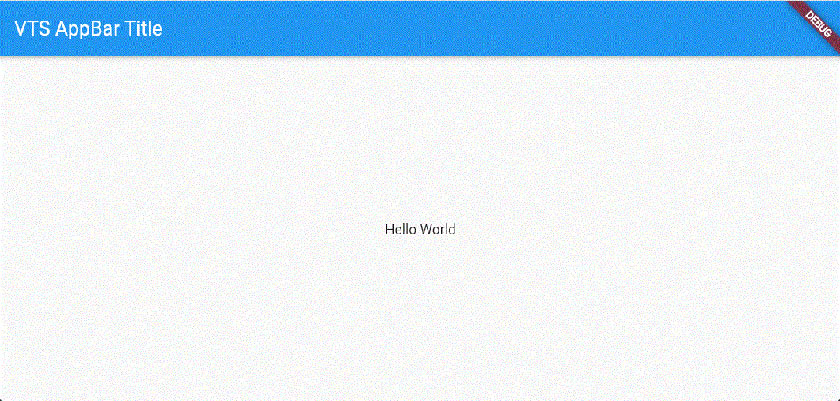
),

);

}

}

Kết quả hiển thị



Căn lề title trong AppBar

Mặc định title sẽ được căn lề bên trái, tuy nhiên bạn có thể căn lại lề theo mong muốn thông qua thuộc tính

* *alignment: Alignment.center  : Căn lề giữa*
* *alignment: Alignment.centerRight : Căn lề bên phải*
* *alignment: Alignment.centerLeft : Căn lề bên trái*

Cú pháp câu lệnh như  sau :

appBar: AppBar(

title: Align (

child: Text("VTS AppBar Title"),

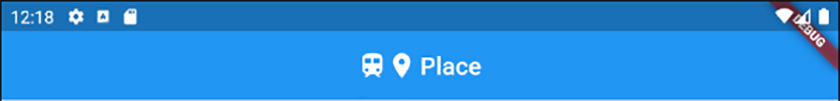
alignment: Alignment.center

)

),

Chèn biểu tượng vào AppBar

Bạn cũng có thể chèn thêm các Icon bên canh title của AppBar, để tạo được icon bạn sẽ phải thêm vào một Widget bao gồm các icon và title



import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

const MyHomePage({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: IconTitleWidget()

),

body: Center(

child: Text(

'VTS chuyên nhận đào tạo lập trình Flutter',

)

),

);

}

}

class IconTitleWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

imageCache?.clear();

return Row (

mainAxisAlignment: MainAxisAlignment.center, // Centers horizontally

crossAxisAlignment: CrossAxisAlignment.center, // Centers vertically

children: [

Icon(Icons.train),

Icon(Icons.place),

// The SizedBox provides an immediate spacing between the widgets

SizedBox (

width: 5,

),

Text(

"VTS Place",

)

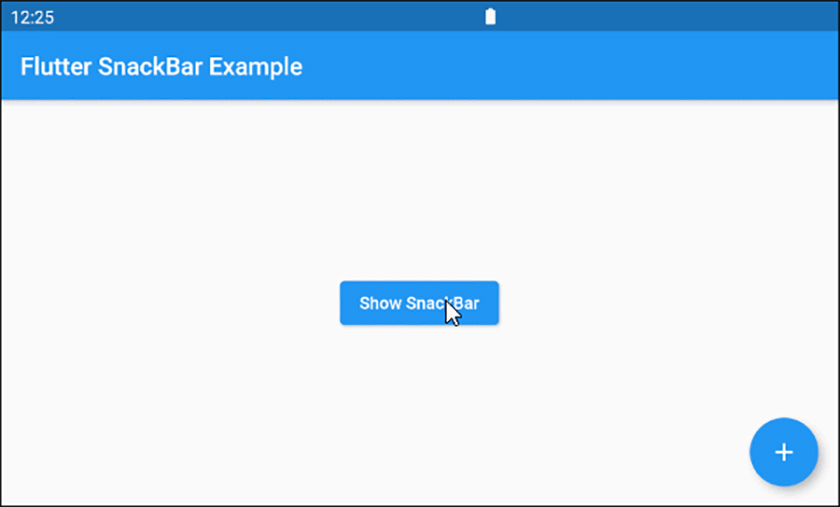
],

);

}

}

Kết quả màn hình



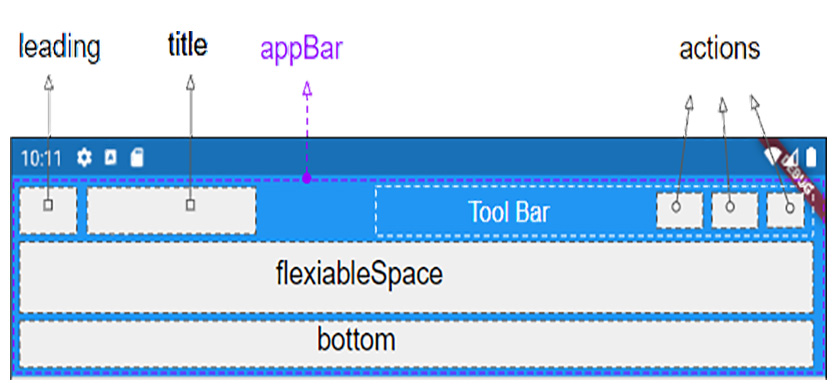
FlexiableSpace

***Flexiablespace*** là một không gian linh hoạt đứng ngay phía dưới AppBar có kích thước chiều cao bằng với chiều cao AppBar

***FlexiableSpace*** dùng để ngăn sách thành phần AppBar với các thành phần khác trong Scraffold

Action

Khu vực action nằm ở phía trên bên phải của Scraffold, dùng để thêm các nút lệnh thực hiện một nhiệm vụ nào đó, nút lệnh (action) thường là các IconButton hoặc PopupMenuButton

Trong ví dụ này chúng ta sẽ thêm 2 nút iconButton và PopupMenuButton

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

const MyHomePage({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS AppBar Title"),

actions: [

IconButton(

icon: Icon(Icons.file\_upload),

onPressed: () => {

print("Click on upload button")

},

),

IconButton(

icon: Icon(Icons.settings),

onPressed: () => {

print("Click on settings button")

}

),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

)

]

),

body: Center(

child: Text(

'VTS nhận đào tạo lập trình Flutter từ A-Z',

)

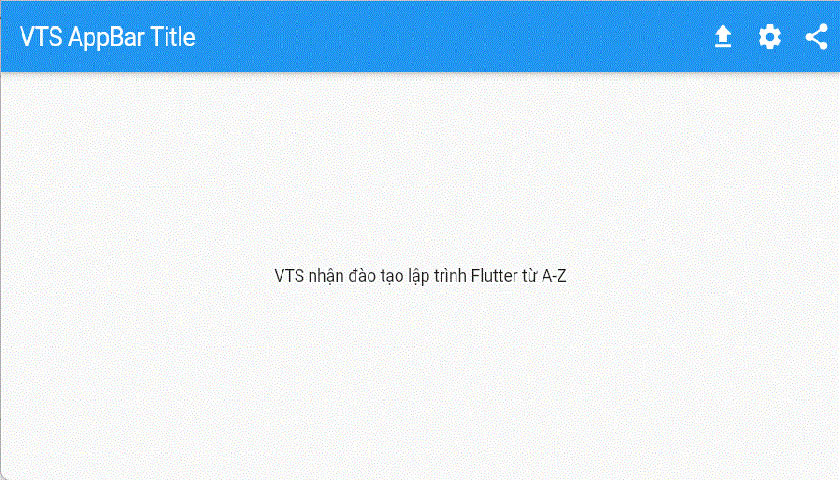
),

);

}

}

kết quả thực hiện



**Bottom**

***Bottom*** (đáy) là khu vực nằm ở phía cuối cùng của AppBar, nó thường được sử dụng để chứa TabBar

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage\_botton(),

);

}

}

class MyHomePage\_botton extends StatelessWidget {

const MyHomePage\_botton({Key? key}) : super(key: key);

Widget build(BuildContext context) {

return DefaultTabController(

length: 3,

child: Scaffold(

appBar: AppBar(

bottom: TabBar(

tabs: [

Tab(icon: Icon(Icons.directions\_car)),

Tab(icon: Icon(Icons.directions\_transit)),

Tab(icon: Icon(Icons.directions\_bike)),

],

),

title: Text('VTS AppBar Example'),

),

body: TabBarView (

children: [

Center(child: Text("Car")),

Center(child: Text("Transit")),

Center(child: Text("Bike")

],

),

)

);

}

}

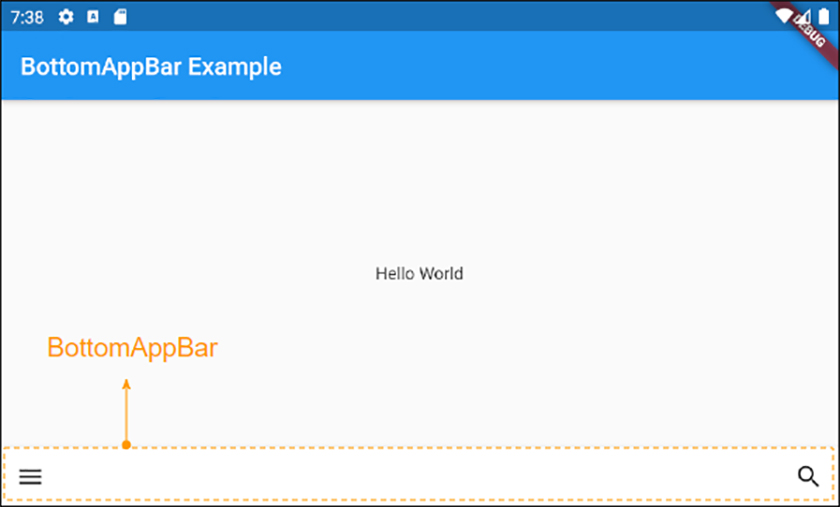
**Bài 5. Tìm hiểu BottomAppBar trong Flutter**

* [BottomAppBar là gì?](https://salesoft.vn/tin-tuc/bottomappbar-flutter#bottomappbar-l%C3%A0-g%C3%AC?)
* [Cấu trúc của BottomAppBar](https://salesoft.vn/tin-tuc/bottomappbar-flutter#c%E1%BA%A5u-tr%C3%BAc-c%E1%BB%A7a-bottomappbar)
* [Child property](https://salesoft.vn/tin-tuc/bottomappbar-flutter#child-property)
* [Shape property](https://salesoft.vn/tin-tuc/bottomappbar-flutter#shape-property)
* [Color property](https://salesoft.vn/tin-tuc/bottomappbar-flutter#color-property)

***BottomAppBar***là gì?

***BottomAppBar***thuộc widget **Scaffold**nằm ở phía cuối của **Scaffold**và được cố định cuối màn hình, nó thường được dùng chứa các chức năng nổi bật của ứng dụng và cho phép người dùng dễ dàng truy cập đến các chức năng này

**BottomAppBar**cho phép gắn với một **FloatingActionButton**



***Cấu trúc của BottomAppBar***

const BottomAppBar(

{Key key,

Color color,

double elevation,

NotchedShape shape,

Clip clipBehavior: Clip.none,

double notchMargin: 4.0,

Widget child}

)

**Child property**

**child**là một thuộc tính quan trọng hầu như được sử dụng trong mọi ***BottomAppBar***, child có thể là một Row dùng để chứa nhiều Widget con như *IconButton*, *PopupMenuButton*,..

Dưới đây là một ví dụ khai báo một ***BottomAppBar***với các Widget *IconButton*và *PopupMenuButton*

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

bottomNavigationBar: BottomAppBar(

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),

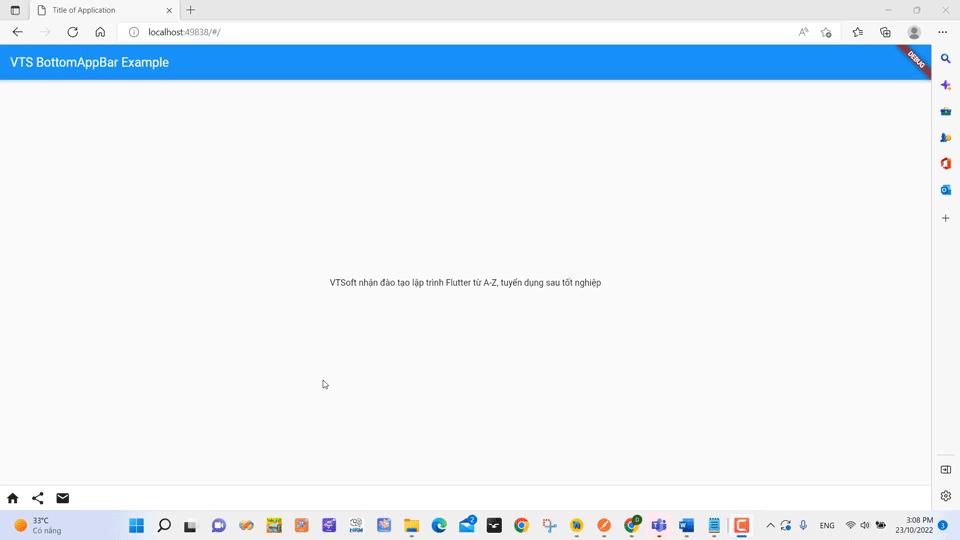
)

);

}

}

Kết quả hiển thị như sau :



Doạn code dưới đây sẽ thêm một ***FloatingActionButton***vào khu vực ***BottomAppBar***

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

floatingActionButton: FloatingActionButton.extended (

elevation: 4.0,

icon: const Icon(Icons.add),

label: const Text('Thêm nhiệm vụ'),

onPressed: () {},

),

floatingActionButtonLocation: FloatingActionButtonLocation.endDocked,

bottomNavigationBar: BottomAppBar(

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),

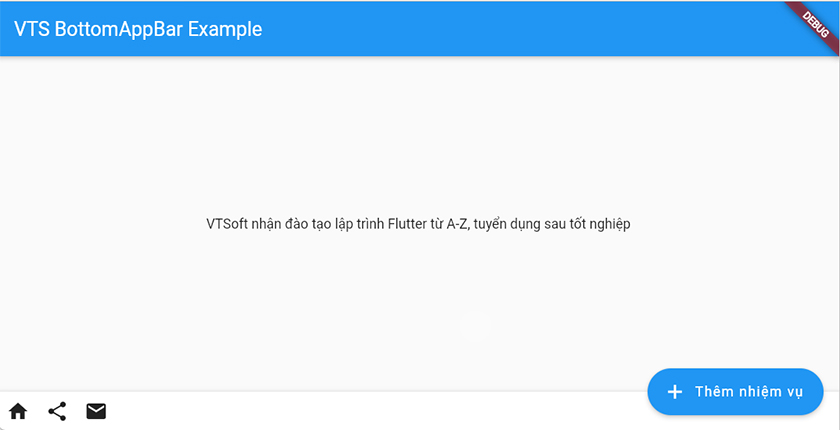
)

);

}

}

Sau khi thêm nút FloatingActionButton và khởi động ứng dụng sẽ có kết quả như sau :



Để neo đối tượng ***FloatingActionButton***vào giữa của ***BottonAppBar***bạn sẽ sử dụng cậu lệnh

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

floatingActionButton: FloatingActionButton(

onPressed: () { },

tooltip: 'Increment',

child: Icon(Icons.add),

elevation: 2.0,

),

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

bottomNavigationBar: BottomAppBar(

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),

)

);

}

}

Để điều chỉnh chiều cao của ***BottomAppBar***bạn sử dụng Widget ***Container***bên ngoài Row như ví dụ sau :

height: 90.0,

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

floatingActionButton: FloatingActionButton(

onPressed: () { },

tooltip: 'Increment',

child: Icon(Icons.add),

elevation: 2.0,

),

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

bottomNavigationBar: BottomAppBar(

child: Container(

height: 90.0,

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.spaceBetween,

children: [

IconButton(icon: Icon(Icons.menu), onPressed: () {},),

IconButton(icon: Icon(Icons.settings), onPressed: () {},),

IconButton(icon: Icon(Icons.search), onPressed: () {},),

],

),

)

/\*child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),\*/

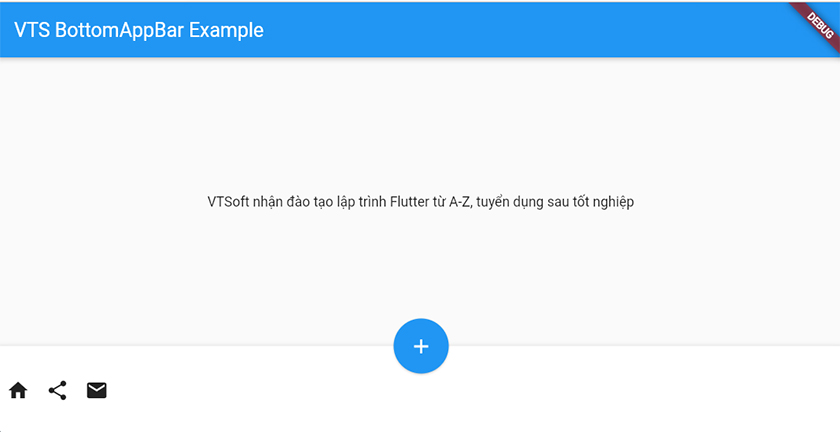
)

);

}

}

Kết quả ứng dụng như sau :



***Shape property***

Sử dụng thuộc tính ***shape***để định dạng vết khắc (notch) khi ***FloatingActionButton***được đặt trên nền của ***BottomAppBar***, chúng tạo ra những đường cong hiệu ứng xung quanh ***FloatingActionButton***

shape: CircularNotchedRectangle(),

Ví dụ đoạn mã đầy đủ về thuộc tính ***shape***

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

floatingActionButton: FloatingActionButton(

onPressed: () { },

tooltip: 'Increment',

child: Icon(Icons.add),

elevation: 2.0,

),

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

bottomNavigationBar: BottomAppBar(

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),

shape: CircularNotchedRectangle(),

)

);

}

}



***Color property***

Thuộc tính ***color***dùng để thay đổi màu nền của ***BottomAppBar***

color: Colors.greenAccent

Ví dụ dưới đây sẽ gán màu nền của ***BottomAppBar***màu xanh lá cây :

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTS BottomAppBar Example"),

),

body: Center(

child: Text(

'VTSoft nhận đào tạo lập trình Flutter từ A-Z, tuyển dụng sau tốt nghiệp',

)

),

floatingActionButton: FloatingActionButton(

onPressed: () { },

tooltip: 'Increment',

child: Icon(Icons.add),

elevation: 2.0,

),

floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,

bottomNavigationBar: BottomAppBar(

child: new Row(

mainAxisSize: MainAxisSize.max,

mainAxisAlignment: MainAxisAlignment.start,

children: [

IconButton(icon: Icon(Icons.home), onPressed: () {},),

PopupMenuButton(

icon: Icon(Icons.share),

itemBuilder: (context) => [

PopupMenuItem(

value: 1,

child: Text("Facebook"),

),

PopupMenuItem(

value: 2,

child: Text("Instagram"),

),

],

),

IconButton(icon: Icon(Icons.email), onPressed: () {},),

],

),

shape: CircularNotchedRectangle(),

color: Colors.greenAccent

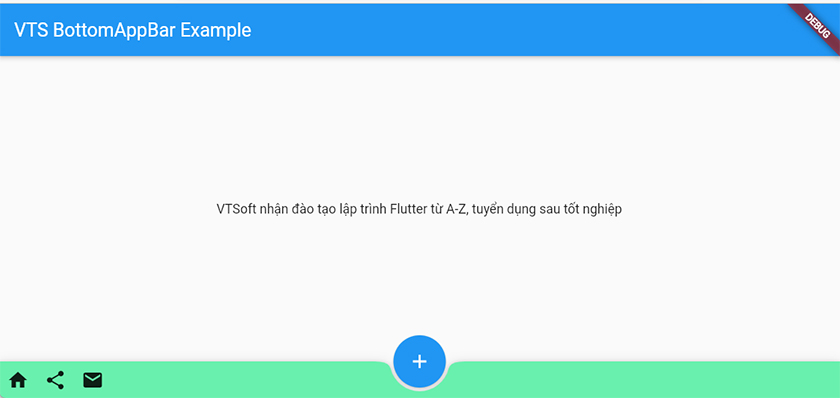
)

);

}

}

Sau khi chạy ứng dụng sẽ nhận được kết quả màu nền như hình dưới đây :



# **Hướng dẫn sử dụng Flutter TabBar và ví dụ**

* [Trang chủ](https://salesoft.vn/)

**Bài 6. Tìm hiểu Tab trong Flutter**

* Tin tức
* [Flutter Tab là gì](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#flutter-tab-l%C3%A0-g%C3%AC)
* [Ví dụ TabBar](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#v%C3%AD-d%E1%BB%A5-tabbar)
* [Cấu trúc TabController, DefaultTabController, TabBar](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#c%E1%BA%A5u-tr%C3%BAc-tabcontroller,-defaulttabcontroller,-tabbar%C2%A0)
* [Các thuộc tính trong Flutter TabBar](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#c%C3%A1c-thu%E1%BB%99c-t%C3%ADnh-trong-flutter-tabbar)
  + [isScrollable](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#isscrollable)
  + [indicatorColor](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#indicatorcolor)
  + [indicatorWeight](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#indicatorweight)
  + [indicatorPadding](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#indicatorpadding)
  + [indicator](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#indicator)
  + [labelColor, unselectedLabelColor](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#labelcolor,%C2%A0unselectedlabelcolor)
  + [labelPadding](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#labelpadding)
  + [labelStyle, unselectedLabelStyle](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#labelstyle,%C2%A0unselectedlabelstyle)
  + [onTap](https://salesoft.vn/tin-tuc/huong-dan-su-dung-flutter-tabbar-va-vi-du#ontap)

## Flutter Tab là gì

***TAB*** là một bố cục giao diện nằm trong đối tượng TabBar, TabBar chứa nhiều Tab và thường được dùng để phân loại các Widget có cùng tính chất với nhau vào cùng một Tab

Tab được sử dụng gắn liền với***TabController, TabBar, TabBarView***

**TabController**

TabController có nhiệm vụ đồng bộ hóa các TAB trong suốt quá trình hoạt động, TabController có thể được tạo ra một cách thủ công hoặc sử dụng Widget có sẵn là DefaultTabController, DefaultTabController hỗ trợ các tính năng cơ bản như khai bản như định nghĩa số lượng các Tab

DefaultTabController(

// The number of tabs / content sections to display.

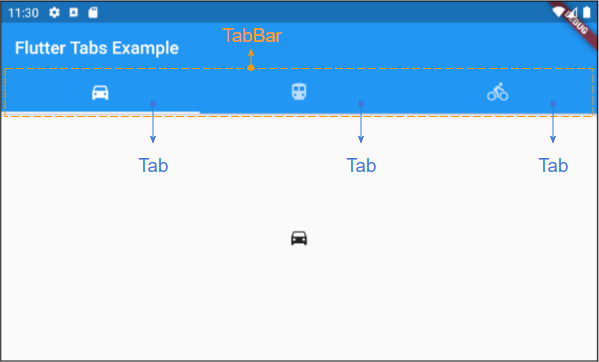
length: 3,

child: // Complete this code in the next step.

);

**TabBar**

TabBar dùng để chứa các Tab, tạo ra các Tab, hãy xem hình dưới đây để có thể hình dung rõ hơn về mối quan hệ giữa TabBar và Tab



DefaultTabController(

length: 3,

child: Scaffold(

appBar: AppBar(

bottom: TabBar(

tabs: [

Tab(icon: Icon(Icons.directions\_car)),

Tab(icon: Icon(Icons.directions\_transit)),

Tab(icon: Icon(Icons.directions\_bike)),

],

),

),

),

);

Trong ví dụ trên chúng ta đã sử dụng DefaultTabController để chứa một TabBar, trong TabBar lại có chứa 3 Tab

Câu hỏi đặt ra là bạn có thể đặt nhiều TabBar trong một DefaultTabController không? Flutter sẽ hỗ trợ bạn việc này tuy nhiên DefaultTabController chỉ làm việc với một TabBar gần với nó nhất, để hoạt động được nhiều TabBar bạn hãy dùng đến TabController được điều khiển bởi chính bạn (điều khiển thủ công)

**TabBarView**

Sử dụng TabBarView để chứa các nội dung tương ứng với mỗi Tab trên TabBar.

TabBarView (

children: [

Icon(Icons.directions\_car),

Icon(Icons.directions\_transit),

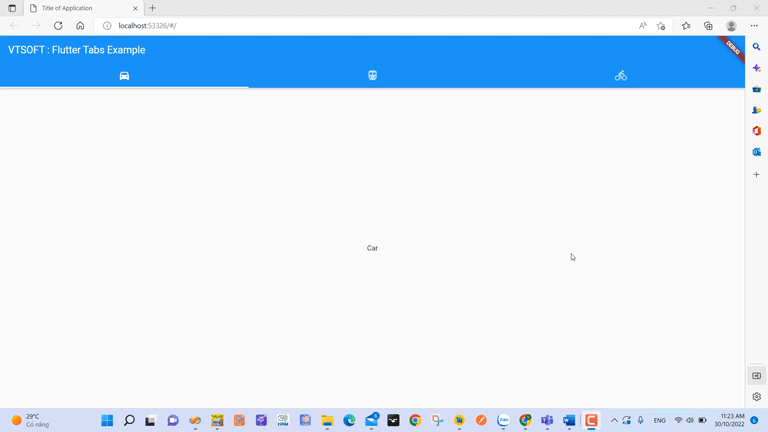
Icon(Icons.directions\_bike),

],

);

## Ví dụ TabBar

Trong ví dụ này chúng ta sẽ tạo ra một TabBar chứa 3 Tab và sử dụng DefaultTabController



## Cấu trúc TabController, DefaultTabController, TabBar

**TabController**

TabController(

{int initialIndex: 0,

@required int length,

@required TickerProvider vsync}

)

**DefaultTabController**

const DefaultTabController(

{Key key,

@required int length,

int initialIndex: 0,

@required Widget child}

)

**TabBar**

const TabBar(

{Key key,

@required List tabs,

TabController controller,

bool isScrollable: false,

Color indicatorColor,

double indicatorWeight: 2.0,

EdgeInsetsGeometry indicatorPadding: EdgeInsets.zero,

Decoration indicator,

TabBarIndicatorSize indicatorSize,

Color labelColor,

TextStyle labelStyle,

EdgeInsetsGeometry labelPadding,

Color unselectedLabelColor,

TextStyle unselectedLabelStyle,

DragStartBehavior dragStartBehavior: DragStartBehavior.start,

MouseCursor mouseCursor,

ValueChanged onTap,

ScrollPhysics physics}

)

**TabBarView**

const TabBarView(

{Key key,

@required List children,

TabController controller,

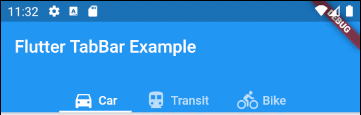
ScrollPhysics physics,

DragStartBehavior dragStartBehavior: DragStartBehavior.start}

)

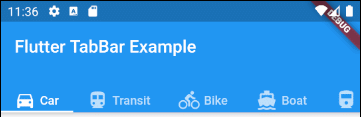
## Các thuộc tính trong Flutter TabBar

### isScrollable

**

Thuộc tính **isScrollable**  có hay không cho phép tạo thanh cuộn ngang khi tổng chiều dài của các tab lớn hơn độ rộng màn hình. khi **isScrollable** được gán là **True** thì độ rộng của mỗi tab chứa vừa đủ nội dung của tab (Hình ảnh và văn bản) và xuất hiện thanh trượt ngang

Bạn hãy xem ví dụ dưới đây



import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

EdgeInsets a2; EdgeInsetsDirectional a;

return DefaultTabController(

length: 6,

child: Scaffold(

appBar: AppBar(

bottom: createTabBar(),

title: Text('Flutter TabBar Example'),

),

body: TabBarView(

children: [

Center(child: Text("Car")),

Center(child: Text("Transit")),

Center(child: Text("Bike")),

Center(child: Text("Boat")),

Center(child: Text("Railway")),

Center(child: Text("Bus Electrical"))

],

),

)

);

}

TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

isScrollable: true,

);

}

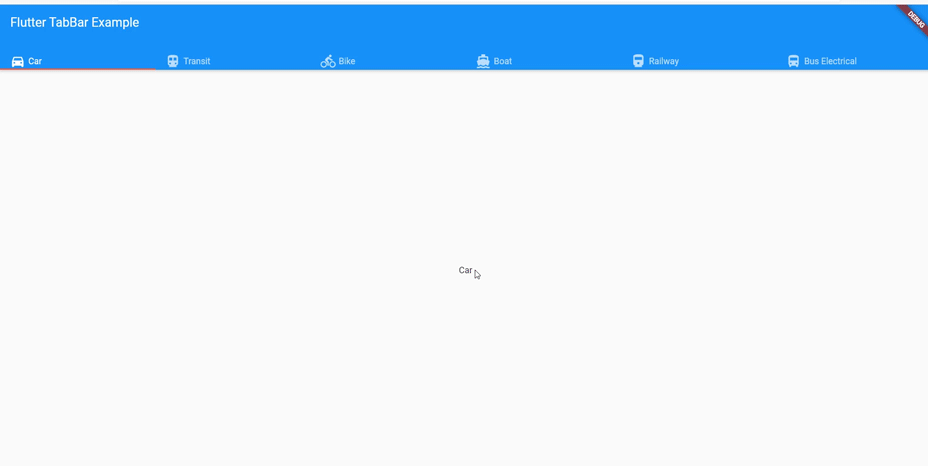
}

### indicatorColor

Thuộc tính indicatorColor dùng để định dạng đường kẻ nền Tab đang được chọn, thuộc tính này sẽ bị vô hiệu lực nếu thuộc tính indicator được sử dụng

indicatorColor: Color(0xffE74C3C),

Ví dụ dưới đây sẽ tạo ra một đường kẻ (Line) màu đỏ khi tab được chọn



import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

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

@override

Widget build(BuildContext context) {

EdgeInsets a2; EdgeInsetsDirectional a;

return DefaultTabController(

length: 6,

child: Scaffold(

appBar: AppBar(

bottom: createTabBar(),

title: Text('Flutter TabBar Example'),

),

body: TabBarView(

children: [

Center(child: Text("Car")),

Center(child: Text("Transit")),

Center(child: Text("Bike")),

Center(child: Text("Boat")),

Center(child: Text("Railway")),

Center(child: Text("Bus Electrical"))

],

),

)

);

}

TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

/\*isScrollable: true,\*/

indicatorColor: Color(0xffE74C3C),

);

}

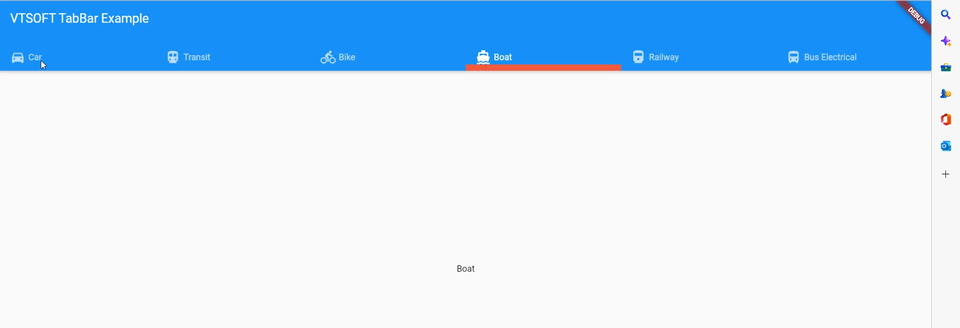
}

### indicatorWeight

Thuộc tính **indicatorWeight** dùng để định dạng độ dày (Thickness) của đường kẻ (Line) phía dưới của **Tab** đang được chon

indicatorWeight: 10

Ví dụ dưới đây sẽ định dạng độ dày của đường kẻ (Line) lên 10 đơn vị khi tab được chọn



TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

/\*isScrollable: true,\*/

indicatorColor: Color(0xffE74C3C),

indicatorWeight: 10,

);

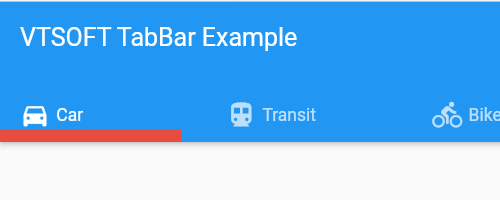
}

}

### indicatorPadding

Thuộc tính **indicatorPadding** dùng để chỉ định khoảng cách đệm (**padding**) của thanh ngang tới mép bên phải của tab

indicatorPadding: EdgeInsets.only(right: 20),



TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

/\*isScrollable: true,\*/

indicatorColor: Color(0xffE74C3C),

indicatorWeight: 10,

indicatorPadding: EdgeInsets.only(right: 20),

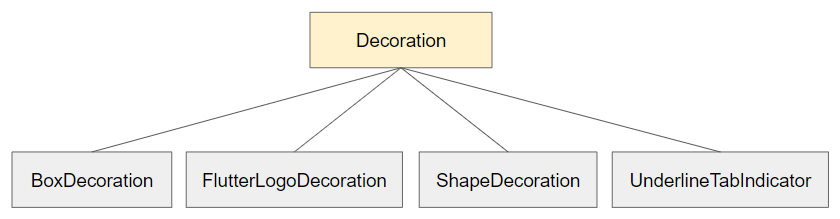
);

}

}

### indicator

Thuộc tính **indicator** dùng để tạo hiệu ức khung cho **Tab** đang được chọn, khi thuộc tính indicator được sử dụng thì các thuộc tính **indicatorColor**, **indicatorWeight** và **indicatorPadding** sẽ bị vô hiệu hóa.



Sau đây là ví dụ cách dùng **indicator**

TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

indicator: ShapeDecoration (

shape: UnderlineInputBorder (

borderSide: BorderSide(

color: Colors.transparent,

width: 0,

style: BorderStyle.solid

)

),

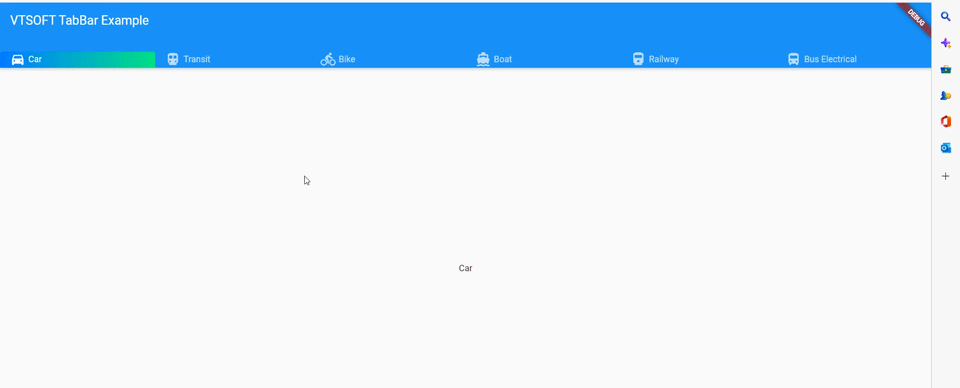
gradient: LinearGradient(colors: [Color(0xff0081ff), Color(0xff01ff80)])

)

);

}

}

****

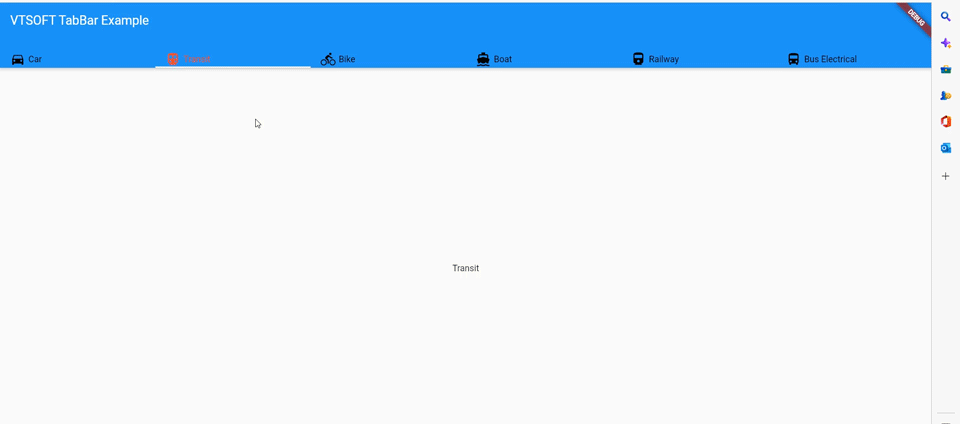
### labelColor, unselectedLabelColor

thuộc tính **labelColor** dùng để chỉ định màu văn bản cho **tab** đang được chọn, thuộc tính **unselectedLabelColor** dùng để chỉ định màu cho các tab không được chọn

Color labelColor;

Color unselectedLabelColor;

Chúng ta cùng tìm hiểu và xem kết quả ở ví dụ sau đây



TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

labelColor: Colors.red,

unselectedLabelColor: Colors.black,

);

}

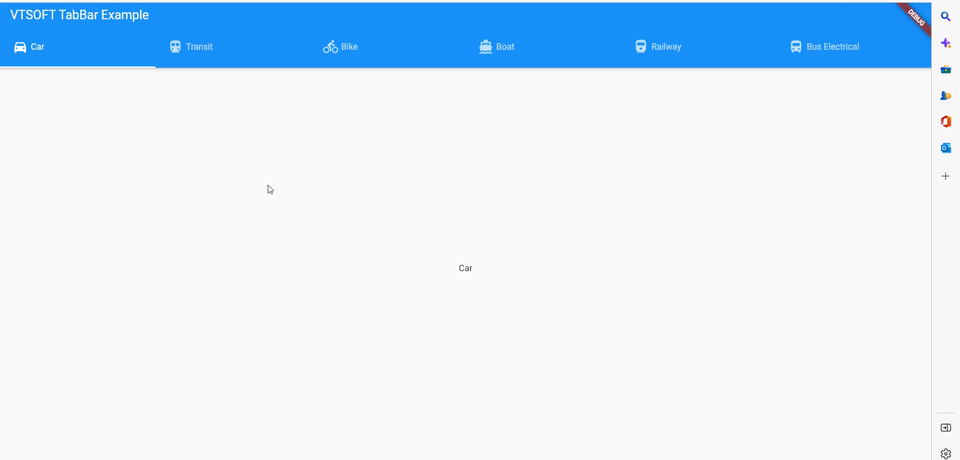
}

### labelPadding

Thuộc tính **labelPadding** dùng để thêm vùng đệm (**padding**) vào cho các lable của từng **tab**

labelPadding: EdgeInsets.all( 20),

Cùng xem ví dụ sau nhé



TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

labelPadding: EdgeInsets.all( 20),

);

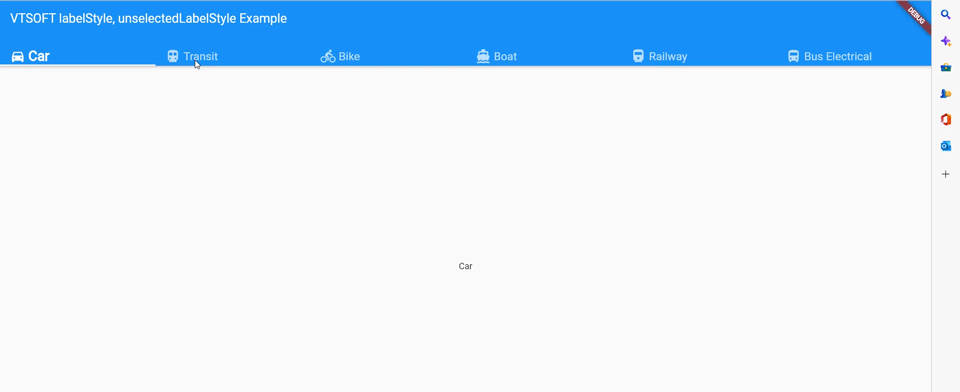
}

}

### labelStyle, unselectedLabelStyle

Thuộc tính **labelStyle**, **unselectedLabelStyle** dùng để định dạng văn bản (lable) khi **tab** đang được chọn hoặc không chọn

Trong vì dụ này lable được chọn sẽ được bôi đậm và có kích thước font size là 20, những lable không được chọn sẽ có font size là 18 và chữ thường



TabBar createTabBar() {

return TabBar(

tabs: [

Row (children: [Icon(Icons.directions\_car), SizedBox(width:5), Text("Car")]),

Row (children: [Icon(Icons.directions\_transit), SizedBox(width:5), Text("Transit")]),

Row (children: [Icon(Icons.directions\_bike), SizedBox(width:5), Text("Bike")]),

Row (children: [Icon(Icons.directions\_boat), SizedBox(width:5), Text("Boat")]),

Row (children: [Icon(Icons.directions\_railway), SizedBox(width:5), Text("Railway")]),

Row (children: [Icon(Icons.directions\_bus), SizedBox(width:5), Text("Bus Electrical")]),

],

labelStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 22),

unselectedLabelStyle: TextStyle(fontStyle: FontStyle.normal, fontSize: 18),

);

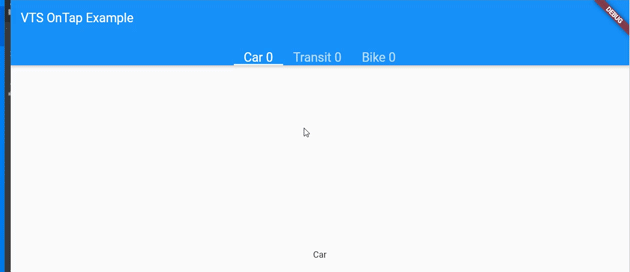
}

}

### onTap

onTap là một sự kiện (hàm) **callback** sẽ được gọi khi người dùng lựa chọn **Tab** trên **TabBar**

Chúng ta sẽ làm một ví dụ khi **Tab** được chọn sẽ đếm số lần **Tab** được chọn và ghi lại trên **lable** của **Tab**

**

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

// This widget is the root of your application.

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

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

@override

State createState() {

return MyHomePageState();

}

}

class MyHomePageState extends State {

int carClick = 0;

int transitClick = 0;

int bikeClick = 0;

@override

Widget build(BuildContext context) {

return DefaultTabController(

length: 3,

child: Scaffold(

appBar: AppBar(

bottom: createTabBar(),

title: Text('VTS OnTap Example'),

),

body: TabBarView(

children: [

Center(child: Text("Car")),

Center(child: Text("Transit")),

Center(child: Text("Bike"))

],

),

)

);

}

TabBar createTabBar() {

return TabBar(

isScrollable: true,

labelStyle: TextStyle(fontSize: 20),

tabs: [

Text("Car " + this.carClick.toString()),

Text("Transit " + this.transitClick.toString()),

Text("Bike " + this.bikeClick.toString())

],

onTap: (index) {

this.onTapHandler(index);

}

);

}

void onTapHandler(int index) {

setState(() {

switch(index){

case 0:

carClick++;

break;

case 1:

transitClick++;

break;

case 2:

bikeClick++;

break;

}

});

}

# **Hướng dẫn và các ví dụ sử dụng Flutter BottomNavigationBar**

* [Trang chủ](https://salesoft.vn/)

* Tin tức

## Bài 7. Thanh điều hướng BottomNavigationBar

* [BottomNavigationBar](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#bottomnavigationbar)
* [Các sự kiện thuộc tính của BottomNavigationBar](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#c%C3%A1c-s%E1%BB%B1-ki%E1%BB%87n-thu%E1%BB%99c-t%C3%ADnh-c%E1%BB%A7a-bottomnavigationbar)
  + [Example](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#example)
  + [items](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#items)
  + [onTap](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#ontap)
  + [currentIndex](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#currentindex)
  + [type](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#type)
  + [iconSize](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#iconsize)
  + [selectedIconTheme](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#selectedicontheme)
  + [unselectedIconTheme](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#unselectedicontheme)
  + [selectedLabelStyle, unselectedLabelStyle](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#selectedlabelstyle,%C2%A0unselectedlabelstyle)
  + [showSelectedLabels showUnselectedLabels](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#showselectedlabels%C2%A0showunselectedlabels)
  + [selectedFontSize unselectedFontSize](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#selectedfontsize%C2%A0unselectedfontsize)
  + [backgroundColor](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#backgroundcolor)
  + [selectedItemColor, unselectedItemColor](https://salesoft.vn/tin-tuc/flutter-bottomnavigationbar#selecteditemcolor,%C2%A0unselecteditemcolor)

**BottomNavigationBar** là một thanh điều hướng (**Navigation**) nằm ở phía cuối của màn hình (**Bottom**), trong các ứng dụng truyền thống BottomNavigationBar dùng chứa một vài chức năng chính và hoặc chứa những chức năng người dùng thường xuyên sử dụng

**BottomNavigationBar** thường được đặt trong một Scaffold thông qua thuộc tính (property) **AppBar.bottomNavigationBar**

**BottomNavigationBar** có tính năng sử dụng khá giống với BottomAppBar, BottomNavigationBar thường được dùng cho các mục đích đơn giản hơn BottomAppBar như chỉ chứa các Button. Nếu bạn muốn có nhiều dạng Widget hoặc muốn sáng tạo hơn thì nên sử dụng BottomAppBar

**BottomNavigationBar Construction**

BottomNavigationBar(

{Key key,

@required List items,

ValueChanged onTap,

int currentIndex: 0,

double elevation,

BottomNavigationBarType type,

Color fixedColor,

Color backgroundColor,

double iconSize: 24.0,

Color selectedItemColor,

Color unselectedItemColor,

IconThemeData selectedIconTheme,

IconThemeData unselectedIconTheme,

double selectedFontSize: 14.0,

double unselectedFontSize: 12.0,

TextStyle selectedLabelStyle,

TextStyle unselectedLabelStyle,

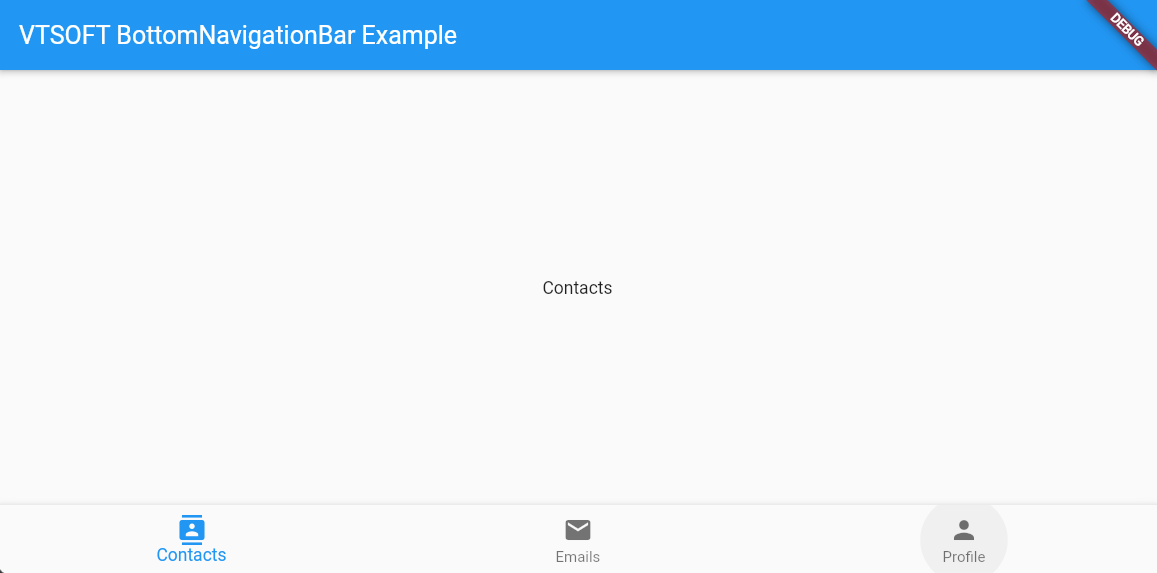
bool showSelectedLabels: true,

bool showUnselectedLabels,

MouseCursor mouseCursor}

)

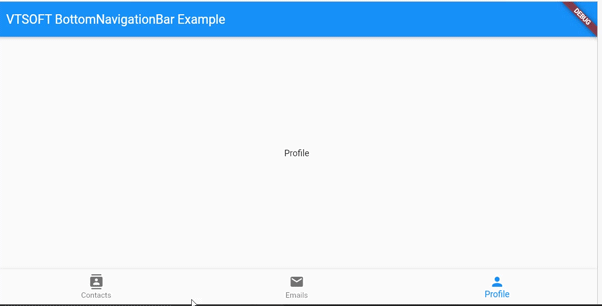
Hãy cùng xem hình ảnh mô tả **BottomNavigationBar**dưới đây



## Các sự kiện thuộc tính của BottomNavigationBar

### Example

Chúng ta hay cùng xem một ví dụ tương đối hoàn chỉnh để hiểu rõ hơn về **BottomNavigationBar** sau đây nhé



import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

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

@override

State createState() {

return MyHomePageState();

}

}

class MyHomePageState extends State {

int selectedIndex = 0;

Widget \_myContacts = MyContacts();

Widget \_myEmails = MyEmails();

Widget \_myProfile = MyProfile();

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTSOFT BottomNavigationBar Example"),

),

body: this.getBody(),

bottomNavigationBar: BottomNavigationBar(

type: BottomNavigationBarType.fixed,

currentIndex: this.selectedIndex,

items: [

BottomNavigationBarItem(

icon: Icon(Icons.contacts),

title: Text("Contacts"),

),

BottomNavigationBarItem(

icon: Icon(Icons.mail),

title: Text("Emails"),

),

BottomNavigationBarItem(

icon: Icon(Icons.person),

title: Text("Profile"),

)

],

onTap: (int index) {

this.onTapHandler(index);

},

),

);

}

Widget getBody( ) {

if(this.selectedIndex == 0) {

return this.\_myContacts;

} else if(this.selectedIndex==1) {

return this.\_myEmails;

} else {

return this.\_myProfile;

}

}

void onTapHandler(int index) {

this.setState(() {

this.selectedIndex = index;

});

}

}

class MyContacts extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(child: Text("Contacts"));

}

}

class MyEmails extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(child: Text("Emails"));

}

}

class MyProfile extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Center(child: Text("Profile"));

}

}

### items

Thuộc tính Items dùng để định nghĩa số các **Item** có trong **BottomNavigationBar**, Số **Items**này phải từ 2 trở lên, nếu bạn khai báo dưới 2 **Item**thì sẽ gặp thông báo lỗi

@required List items

**BottomNavigationBarItem** Construction

const BottomNavigationBarItem(

{@required Widget icon,

Widget title,

Widget activeIcon,

Color backgroundColor}

)

Đoạn mã khai báo Items

items: [

BottomNavigationBarItem(

icon: Icon(Icons.contacts),

title: Text("Contacts"),

),

BottomNavigationBarItem(

icon: Icon(Icons.mail),

title: Text("Emails"),

),

BottomNavigationBarItem(

icon: Icon(Icons.person),

title: Text("Profile"),

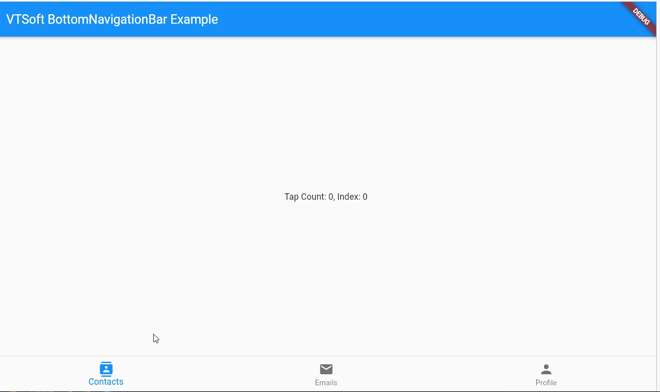
)

],

### onTap

**OnTap** là một hàm **callback** và đượcthực thi khi người dùng chạm (**tap**) và một **item** của **BottomNavigationBar**.

Chúng ta hãy cùng theo dõi ví dụ sau đây, khi chạm vào một Item ứng dụng sẽ đếm số lần chạm (**tap**) và **item** và ghi ra **index** của **item** đang thực thi



import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

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

@override

State createState() {

return MyHomePageState();

}

}

class MyHomePageState extends State {

int tapCount = 0;

int selectedIndex = 0;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTSoft BottomNavigationBar Example"),

),

body: Center(

child: Text("Tap Count: " + this.tapCount.toString()

+ ", Index: " + this.selectedIndex.toString())

),

bottomNavigationBar: BottomNavigationBar(

currentIndex: this.selectedIndex,

items: [

BottomNavigationBarItem(

icon: Icon(Icons.contacts),

title: Text("Contacts"),

),

BottomNavigationBarItem(

icon: Icon(Icons.mail),

title: Text("Emails"),

),

BottomNavigationBarItem(

icon: Icon(Icons.person),

title: Text("Profile"),

)

],

onTap: (int index) {

this.onTapHandler(index);

},

),

);

}

void onTapHandler(int index) {

this.setState(() {

this.tapCount++;

this.selectedIndex = index;

});

}

}

### currentIndex

Thuộc tính **currentIndex** dùng để xác định vị trí **Item** đang được chọn trong **BottomNavigationBar**, chỉ số currentIndex được bắt đầu từ vị trí 0 tương đương **Item** đầu tiên trong **BottomNavigationBar**

int currentIndex: 0

### type

Thuộc tính **type**để cấu hình hiển thị kiểu của **BottomNavigationBar**

BottomNavigationBarType type

### iconSize

Thuộc tính **iconSize** dùng để xác định kích thước của biểu tượng của tất cả các **BottomNavigationBar**

double iconSize: 24.0



### selectedIconTheme

Thuộc tính **selectedIconTheme** được sử dụng để thiết lập kích thước (**Size**), màu xắc (**Color**) và độ mờ đục (**opacity**) của biểu được **BottomNavigationBar** đang được lựa chọn

IconThemeData selectedIconTheme

Cấu trúc lệnh như sau :

const IconThemeData (

{Color color,

double opacity,

double size}

)

Bên cạnh thuộc tính selectedIconTheme thì bạn nên sử dụng thêm thuộc tính unselectedIconTheme để định dạng những biểu tượng không được chọn trong BottomNavigationBar để trả trả lại trạng thái bình thường cho những

Hãy cùng xem ví dụ dưới đây

Flutter BottomNavigationBar selectedIconTheme

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Title of Application',

theme: ThemeData(

primarySwatch: Colors.blue,

visualDensity: VisualDensity.adaptivePlatformDensity,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

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

@override

State createState() {

return MyHomePageState();

}

}

class MyHomePageState extends State {

int tapCount = 0;

int selectedIndex = 0;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("VTSoft BottomNavigationBar Example"),

),

body: Center(

child: Text("Tap Count: " + this.tapCount.toString()

+ ", Index: " + this.selectedIndex.toString())

),

bottomNavigationBar: BottomNavigationBar(

currentIndex: this.selectedIndex,

selectedIconTheme: IconThemeData (

color: Colors.red,

opacity: 1.0,

size: 30

),

items: [

BottomNavigationBarItem(

icon: Icon(Icons.contacts),

title: Text("Contacts"),

),

BottomNavigationBarItem(

icon: Icon(Icons.mail),

title: Text("Emails"),

),

BottomNavigationBarItem(

icon: Icon(Icons.person),

title: Text("Profile"),

)

],

onTap: (int index) {

this.onTapHandler(index);

},

),

);

}

void onTapHandler(int index) {

this.setState(() {

this.tapCount++;

this.selectedIndex = index;

});

}

}

### unselectedIconTheme

Thuộc tính **unselectedIconTheme** được sử dụng để thiết lập kích thước (**size**), mầu sắc (**color**) và độ mờ đục (**opacity**) cho biểu tượng của các **BottomNavigationBarItem** không được chọn

IconThemeData unselectedIconTheme

Sau đây là cấu trúc của **unselectedIconTheme**

const IconThemeData (

{Color color,

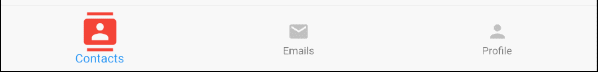
double opacity,

double size}

)

Khi thuộc tính **unselectedIconTheme** được sử dụng, bạn cũng nên chỉ định giá trị cho property **selectedIconTheme**, nếu không bạn sẽ không nhìn thấy biểu tượng trên **BottomNavigationBarItem** đang được chọn.

Ví dụ dưới đây sẽ kết hợp thuộc tính **selectedIconTheme** và **unSelectedIconTheme** để hiển thị kích thước, kiểu chữ, và màu sắc khác biệt khi Item được chọn và không được chọn



selectedIconTheme: IconThemeData (

color: Colors.red,

opacity: 1.0,

size: 45

),

unselectedIconTheme: IconThemeData (

color: Colors.black45,

opacity: 0.5,

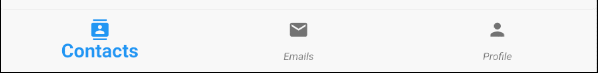
size: 25

),

### selectedLabelStyle, unselectedLabelStyle

thuộc tính **selectedLabelStyle** và **unselectedLabelStyle** sử dụng để thiết lập kiểu dáng chữ cho văn bản khi  **BottomNavigationBarItem** được chon hoặc không được chọn

cùng xem ví dụ sau



selectedLabelStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 22),

unselectedLabelStyle: TextStyle(fontStyle: FontStyle.italic),

### showSelectedLabels showUnselectedLabels

thuộc tính **showSelectedLabels**và **showUnselectedLabels**dùng để cho phép hoặc không cho phép hiển thị văn bản khi BottomNavigationBarItem được chọn hoặc không được chọn

bool showSelectedLabels: true

bool showUnselectedLabels : true

Hãy xem ví dụ sau đây :

BottomNavigation showSelectedLabels

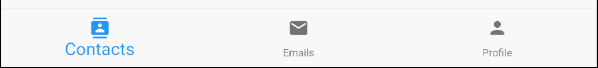
showSelectedLabels: true,

showUnselectedLabels: false,

### selectedFontSize unselectedFontSize

Thuộc tính **selectedFontSize**, **unselectedFontSize**dùng để thiết lập kích cỡ font chữ cho văn bản của **BottomNavigationBarItem**đang được chọn và đang không được chọn

Chúng ta cùng theo dõi ví dụ sau đây :



selectedFontSize: 20,

unselectedFontSize: 15,

### backgroundColor

Thuộc tính **backgroundColor** được sử dụng để chỉ định mầu nền cho **BottomNavigationBar**.

BottomNavigationBar backgroundColor

backgroundColor : Colors.greenAccent,

### selectedItemColor, unselectedItemColor

Thuộc tính **selectedItemColor** dùng để chỉ định văn bản và biểu tượng cho **BottomNavigationBarItem** được chọn

**unselectedItemColor** dùng để chỉ định màu văn bản và và biểu tượng cho những **bottomNavigationbarItem** không được chọn

Chúng ta sử dụng đồng thời 2 sự kiện này để tạo cảm nhận cho người dùng rõ hiệu ứng trên **bottomNavigationbarItem**

BottomNavigation selectedItemColor

selectedItemColor : Colors.red,

unselectedItemColor: Colors.cyan,

# **So sánh StatelessWidget và StatefulWidget**

## 1. Intro

Xin chào các bạn, đây là bài viết đầu tiên của mình trên blog của cộng đồng **200lab**. Trước đây mình cũng đã có một số bài viết khác về Flutter và được đăng trên github cá nhân của mình. Mọi người có thể follow tại [đây](https://github.com/vanle57?ref=200lab.io). Bây giờ thì bắt đầu thôi nào!

## 2. Mở đầu

Khi tiếp xúc với Flutter, có lẽ các bạn đều đã quen hoặc nếu không quen thì cũng đã từng nghe câu “Everything is a widget!” – có nghĩa là tất cả mọi thứ đều là widget. Text là một widget, Button là một Widget, các Animation cũng là Widget, thậm chí bản thân app cũng là một Widget. Nghe nhiều là vậy nhưng thực chất các widget này lại được chia thành hai nhóm chính là StalessWidget và StatefulWidget.

Mình cũng khá chắc là trong số những bạn đang đọc bài viết này cũng có một số bạn đã biết về hai loại này nhưng để hiểu về nó thì còn khá mơ hồ. Vậy nên bài viết này mình sẽ cố gắng trình bày về chúng một cách đơn giản nhất có thể để các bạn nắm bắt được.

## 3. Chuẩn bị

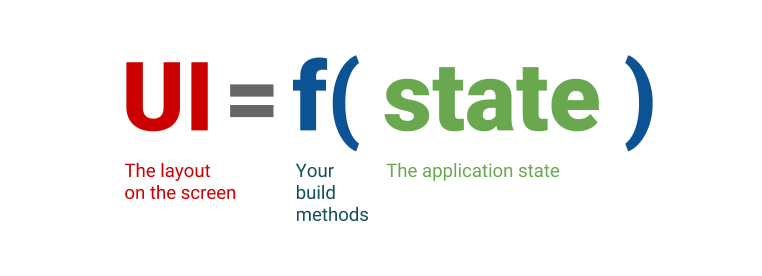
* IDE:
  + Visual Studio Code version 1.67.0
  + Android Studio Chipmunk 2021.2.1
  + XCode version 13.3.1
* Flutter SDK version 2.10.5
* Kiến thức: ngôn ngữ lập trình Dart và Flutter cơ bản

## 4. Khái niệm State

**State** trong tiếng Anh có nghĩa là trạng thái. Trong thế giới của mobile dev, nó thể hiện các trạng thái của ứng dụng. Mình ví dụ đơn giản nhé. Khi các bạn check / uncheck check box thì đó là hai trạng thái của check box. Ví dụ rõ ràng hơn nữa là màn hình đăng nhập thể hiện ba trạng thái nhập email / mật khẩu sai định dạng (validate error), đăng nhập thành công hoặc đăng nhập fail (api error).

Nếu các bạn đã từng biết về lập trình mobile với các ngôn ngữ truyền thống như Swift hay Kotlin thì bây giờ chính là lúc chúng ta thay đổi tư duy!

Flutter chuyển đổi cách lập trình truyền thống từ lập trình mệnh lệnh (imperative) sang lập trình khai báo (declarative). Điều này có nghĩa là Flutter UI phản ánh các trạng thái trong app:



Trong định nghĩa của trang Flutter ở [đây](https://api.flutter.dev/flutter/widgets/State-class.html?ref=200lab.io) có nhắc đến, State có thể được đọc một cách đồng bộ khi widget được build và có thể thay đổi trong suốt vòng đời của widget đó.

Vậy thì Stateless Widget và Stateful Widget là gì? Chúng khác nhau ra sao? Chúng ta cùng tìm hiểu tiếp nhé!

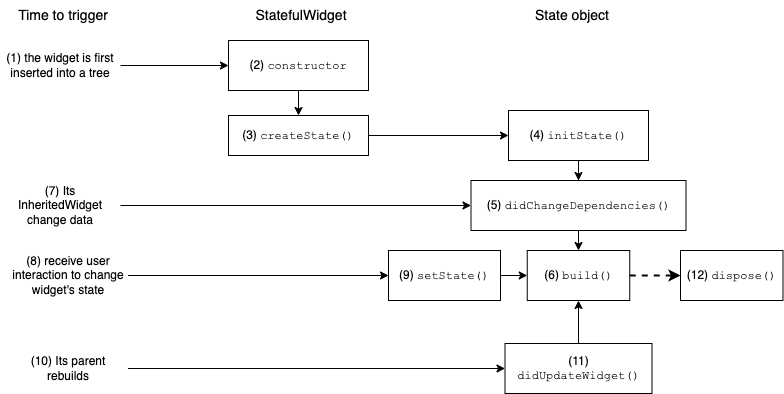
## 5. StatelessWidget vs StatefulWidget

### 5.1. StatelessWidget

* **Stateless widget** là các widget có trạng thái không thể thay đổi sau khi chúng được build. Chúng chỉ đơn thuần nhận dữ liệu và hiển thị một cách thụ động. Nếu muốn render lại thì ta phải khởi tạo lại chúng.
* Một ví dụ về stateles widget là ***Text***. Nó chỉ đơn giản hiển thị các văn bản mà chúng  
  ta truyền cho nó. Ví dụ khác là ***IconButton***. Nó chỉ khác Text ở phần là chúng ta có thể tương tác với button nhưng điều đó cũng không làm thay đổi bất cứ điều gì ở button.
* Vòng đời: Vì không thay đổi nên vòng đời của stateless widget khá đơn giản, chỉ xoay quanh hàm [build()](https://api.flutter.dev/flutter/widgets/State/build.html?ref=200lab.io). Câu hỏi được đặt ra ở đây là hàm build() của chúng sẽ được gọi khi nào? Mình có thể liệt kê ra những thời điểm như sau:
  + Lần đầu tiên widget được khởi tạo và chèn vào trong widget tree. Tìm hiểu thêm về widget tree tại [Flutter cơ bản: Widget Tree, Element Tree và Render Tree](https://200lab.io/blog/flutter-co-ban-widget-tree-element-tree-render-tree/).
  + Khi widget cha thay đổi cấu hình thì cũng sẽ kích hoạt hàm build() của các widget con và khiến chúng được render lại.
  + Khi InheritedWidget chúng phụ thuộc vào thay đổi. Cái này nghe khá khó hiểu nhỉ? Mình sẽ có một bài viết giải thích cụ thể về InheritedWidget sau nhé!

### 5.2. StatefulWidget

* **Stateful widget** là các widget có thể thay đổi trạng thái. Khác với stateless widget, chúng ta không cần thiết phải khởi tạo lại stateful widget để thay đổi chúng mà chỉ cần thay đổi state của chúng và Flutter sẽ gọi hàm build() để render lại UI cho phù hợp với state đó.
* Nghe hơi mơ hồ phải không? Mình ví dụ như nút like của facebook. Khi user bấm like thì nó sẽ đổi thành màu xanh phải không? Và khi user click thêm một lần nữa thì nó chuyển lại màu xám. Đó chính xác là một stateful widget.
* Vòng đời: Vòng đời của Stateful widget thì khá phức tạp, các bạn hãy theo dõi hình sau đây nhé!



***Giải thích:***

* Trước khi đi vào giải thích trên từng con số ở ảnh thì mình muốn mọi người lưu ý một chút. Một statefut widget sẽ bao gồm hai phần là một widget và một state object. Bản thân đối tượng StatefulWidget là **bất biến (immutable)** và lưu trữ trạng thái **có thể thay đổi (mutable)** của chúng trong State object.
* Trên hình mình chia ra làm 3 phần là:
  + Time to trigger: thời điểm kích hoạt
  + StatefulWidget: phần widget mà mình nhắc đến của StatefulWidget
  + State object: phần trạng thái có thể thay đổi được của StatefulWidget
* Bây giờ cùng phân tích những con số trên hình nào!!!
  + (1): Thời điểm đầu tiên widget được khởi tạo và chèn vào trong widget tree.
  + (2) và (3): Sau khi gọi constructor, Flutter sẽ gọi hàm [createState()](https://api.flutter.dev/flutter/widgets/StatefulWidget/createState.html?ref=200lab.io) để tạo ra đối tượng của State cho widget. Kể từ đó về sau, Flutter đa số chỉ làm việc State object mà thôi (vì đối tượng của StatefulWidget là bất biến, như mình đã nói ở trên).
  + (4): Trên State object, sau khi gọi hàm createState() thì hàm [initState()](https://api.flutter.dev/flutter/widgets/State/initState.html?ref=200lab.io) sẽ được gọi. Và hàm này chỉ được gọi **chỉ một lần duy nhất** cho mỗi 1 đối tượng của State. Bạn có thể override phương thức này để khởi tạo các thuộc tính, khởi tạo dữ liệu, đăng ký Streams, ChangeNotifiers, vân vân và mây mây....
  + (5): Phương thức [didChangeDependencies()](https://docs.flutter.io/flutter/widgets/State/didChangeDependencies.html?ref=200lab.io) được gọi ngay sau initState lần đầu tiên.
  + (6): Hàm build() được gọi để render UI. Vậy là hoàn thành các bước kể từ lần đầu tiên widget được khởi tạo. Sau đây sẽ là những bước thực hiện khi có bất kỳ sự thay đổi nào.
  + (7): Trường hợp InheritedWidget mà widget có dữ liệu phụ thuộc vào thay đổi thì sẽ kích hoạt hàm didChangeDependencies(), sau đó Flutter sẽ gọi build() để render lại UI.
  + (8) và (9): Giả sử như user tác động lên UI để thay đổi state của widget (ví dụ như like / unlike) thì khi đó việc của các developer chúng ta sẽ là gọi hàm [setState()](https://api.flutter.dev/flutter/widgets/State/setState.html?ref=200lab.io) để thay đổi trạng thái của object. Việc này sẽ giúp cho framework được thông báo về sự thay đổi này và sẽ gọi hàm build() để render lại UI tương ứng.
  + (10) và (11): Khi widget cha của nó build lại và yêu cầu các widget con build cập nhật lại, Flutter sẽ thực hiện việc gọi hàm [didUpdateWidget()](https://api.flutter.dev/flutter/widgets/State/didUpdateWidget.html?ref=200lab.io) . Sau đó, hàm build() cũng sẽ được gọi.
  + (12): Hàm [dispose()](https://api.flutter.dev/flutter/widgets/State/dispose.html?ref=200lab.io) được gọi khi state object bị xoá khỏi widget tree vĩnh viễn.

Tóm lại, như bạn đã thấy ở trên hàm build() là cái hàm được gọi liên tục. Bởi vì sao? Bởi vì nó là nơi quyết định UI, nên khi có bất kỳ sự thay đổi cần thiết nào trên UI, nó sẽ được gọi lại để render tương ứng.

## 6. Cách làm việc với StatlessWidget và StatefulWidget

Ý tưởng của mình như sau:

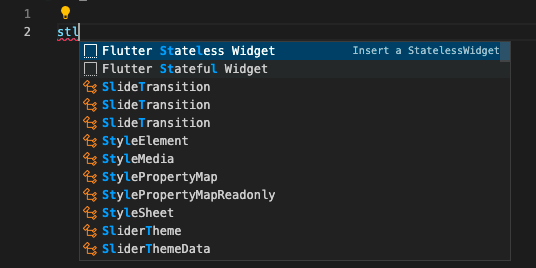
Đầu tiên mình sẽ tạo một button là StalessWidget, sau đó convert button đó sang StatefulWidget. Mục đích của button này sẽ là like / unlike một cái gì đó giống như nút like trên facebook. Okay! Bắt tay vào thôi nào.

### 6.1. StatelessWidget

Các bạn tạo mới Flutter project và file like\_button.dart nhé!

#### Bước 1: Tạo mới một StatelessWidget

Trên VS Code, để tạo mới một StalessWidget bạn chỉ cần gõ stl thì trình biên dịch sẽ đề xuất cho bạn



Bạn đặt tên class là LikeButton và nhớ import package:flutter/material.dart nhé!

dart

copy

import 'package:flutter/material.dart';

class LikeButton extends StatelessWidget {

const LikeButton({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return IconButton(

onPressed: () {

print('Liked');

},

iconSize: 20,

icon: const Icon(Icons.thumb\_up\_off\_alt));

}

}

Action của mình hiện tại chỉ đơn giản là print. Vậy là xong phần LikeButton. Bạn tiếp tục vào file main.dart để sử dụng button này nhé!

#### Bước 2: Sử dụng StatelessWidget đã tạo ở bước 1

Bạn hãy xoá phần class MyHomePage do framework cung cấp sẵn và tự tạo MyHomePage của mình. Vẫn là một StatelessWidget nhé!

dart

copy

class MyHomePage extends StatelessWidget {

const MyHomePage({Key? key}) : super(key: key);

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('StatelessWidget Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: const [

LikeButton(),

Text('You have not like yet!'),

],

),

),

);

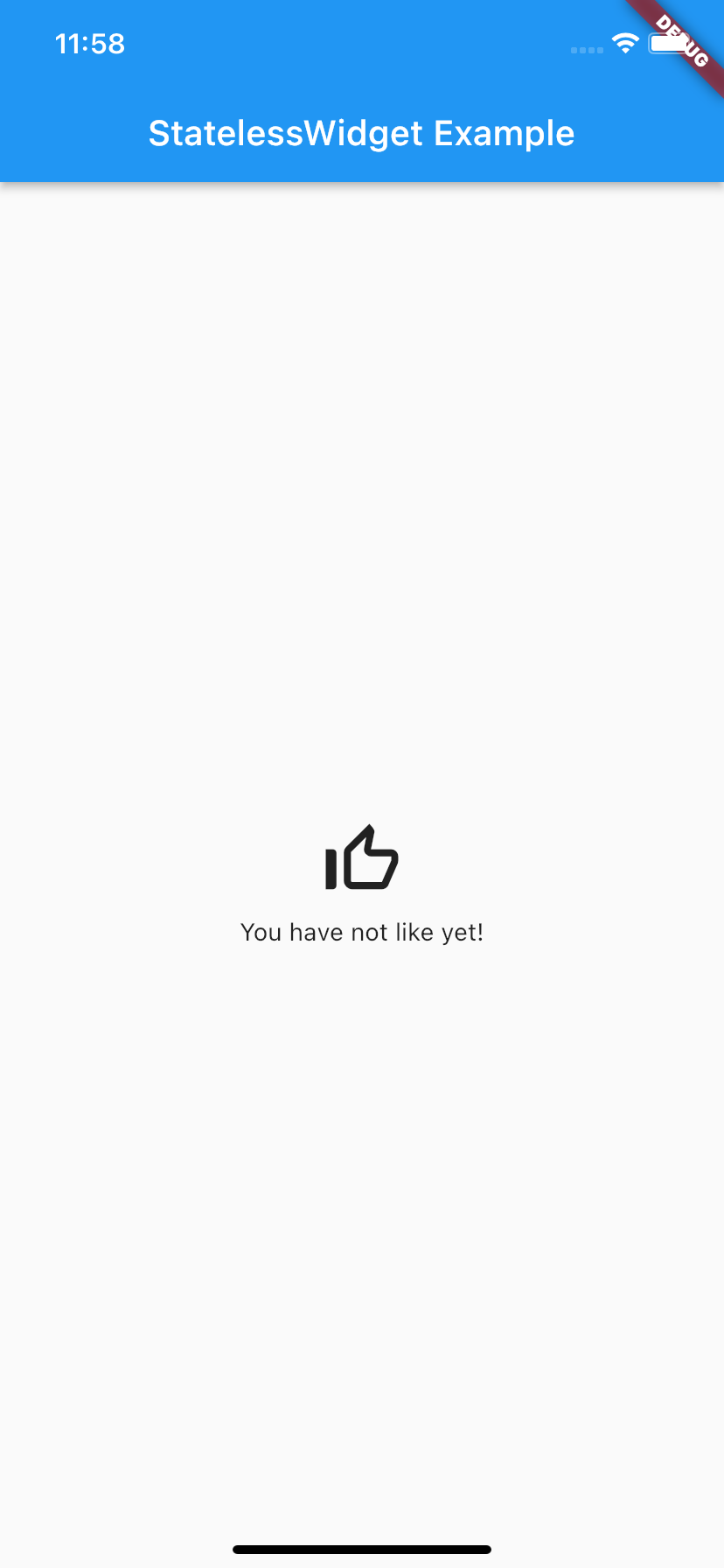
}

}

\****Lưu ý nho nhỏ:*** Ở đây mình có thêm 1 Text để thông báo user đã like hay chưa nhé!

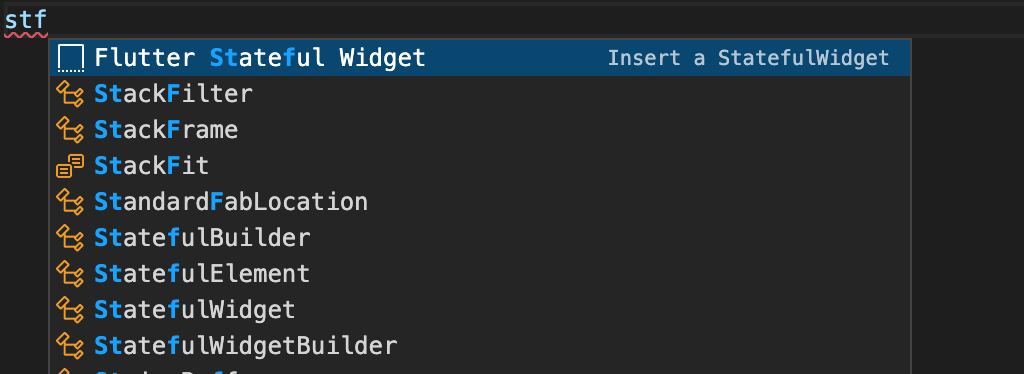
Rồi chúng ta cùng build và xem kết quả nhé!

#### Kết quả:



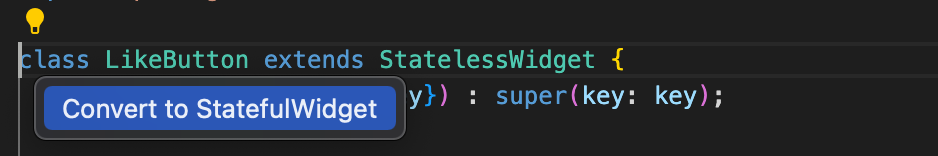
Khi ấn nút like thì trên console sẽ log ra cho ta text "Liked".

### 6.2. StatefulWidget

Các bạn có thể dùng lại project trước đó hoặc tạo mới project tương tự như vậy.  


#### Bước 1: Tạo mới một StatefulWidget

Trên VS Code, để tạo mới một StalefulWidget bạn chỉ cần gõ stf thì trình biên dịch sẽ đề xuất cho bạn. Nếu bạn đang sử dụng project trước thì bạn có thể convert LikeButton sang StatefulWidget bằng cách di chuyển con trỏ về đầu dòng class LikeButton -> click vào bóng đèn màu vàng chọn Convert to StatefulWidget.



Sau khi convert thì LikeButton đã trở thành như thế này:

dart

copy

class LikeButton extends StatefulWidget {

const LikeButton({Key? key}) : super(key: key);

@override

State<LikeButton> createState() => \_LikeButtonState();

}

class \_LikeButtonState extends State<LikeButton> {

@override

Widget build(BuildContext context) {

return IconButton(

onPressed: () {

print('Liked');

},

iconSize: 50,

icon: const Icon(Icons.thumb\_up\_off\_alt));

}

}

#### Bước 2: Xử lý State object

Để biết được user đã like hay chưa thì mình cần có một biến lưu lại trạng thái này.

dart

copy

class \_LikeButtonState extends State<LikeButton> {

// 1

bool isLiked = false;

@override

Widget build(BuildContext context) {

return IconButton(

onPressed: () {

setState(() {

// 3

isLiked = !isLiked;

});

},

iconSize: 50,

// 2

icon: Icon(isLiked ? Icons.thumb\_up\_alt : Icons.thumb\_up\_off\_alt));

}

}

***Giải thích:***

1. Mình sẽ tạo ra biến isLiked kiểu bool với giá trị true nghĩa là user đã like và false là chưa like. Ở đây mình để giá trị default là false.
2. Thay đổi ảnh của button tương ứng với trạng thái isLiked.
3. Khi user tương tác với button, mình sẽ thực hiện việc đảo trạng thái của biến isLiked và thông báo cho framework thông qua việc gọi hàm setState()

Vậy là xong! Chúng ta cùng build lên xem kết quả nhé! Hiện tại, mình vẫn giữ MyHomePage giống như phần StatelessWidget nha!



Các bạn có để ý thấy phần Text ở dưới vẫn chưa khớp với trạng thái của button không? Đây là lúc chúng ta sẽ biến MyHomePage thành một StatefulWidget luôn!

#### Bước 3: Chuyển đổi MyHomePage thành StatefulWidget.

Tương tự như mình đã nhắc ở bước 1, các bạn có thể convert MyHomePage thành StatefulWidget bằng cách di chuyển con trỏ về đầu dòng class MyHomePage -> click vào bóng đèn màu vàng chọn Convert to StatefulWidget.

Tiếp theo, các bạn quay trở lại class LikeButton để thêm callback về nhằm thông báo cho MyHomePage biết trạng thái isLiked

dart

copy

import 'package:flutter/material.dart';

class LikeButton extends StatefulWidget {

// 1

final Function(bool) didTapButton;

// 2

const LikeButton(this.didTapButton, {Key? key}) : super(key: key);

@override

State<LikeButton> createState() => \_LikeButtonState();

}

class \_LikeButtonState extends State<LikeButton> {

bool isLiked = false;

@override

Widget build(BuildContext context) {

return IconButton(

onPressed: () {

setState(() {

isLiked = !isLiked;

// 3

widget.didTapButton(isLiked);

});

},

iconSize: 50,

icon: Icon(isLiked ? Icons.thumb\_up\_alt : Icons.thumb\_up\_off\_alt));

}

}

***Giải thích:***

1. didTapButton là 1 biến kiểu function và trả về kiểu bool. Biến được khai báo là final vì nó sẽ không thay đổi.
2. Tạo constructor để init biến didTapButton.
3. Sau khi isLiked được đảo trạng thái, chúng ta đồng thời callback trạng thái này ra để MyHomePage nhận biết sự thay đổi.

Sau khi sửa lại LikeButton, bạn quay lại MyHomePage và tiếp tục xử lý phần thay đổi Text:

dart

copy

class \_MyHomePageState extends State<MyHomePage> {

// 1

String likedStatusText = 'You have not like yet!';

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('StatefulWidget Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.center,

children: [

// 3

LikeButton((isLiked) {

// 4

setState(() {

likedStatusText =

isLiked ? 'You liked!' : 'You have not like yet!';

});

}),

// 2

Text(likedStatusText),

],

),

),

);

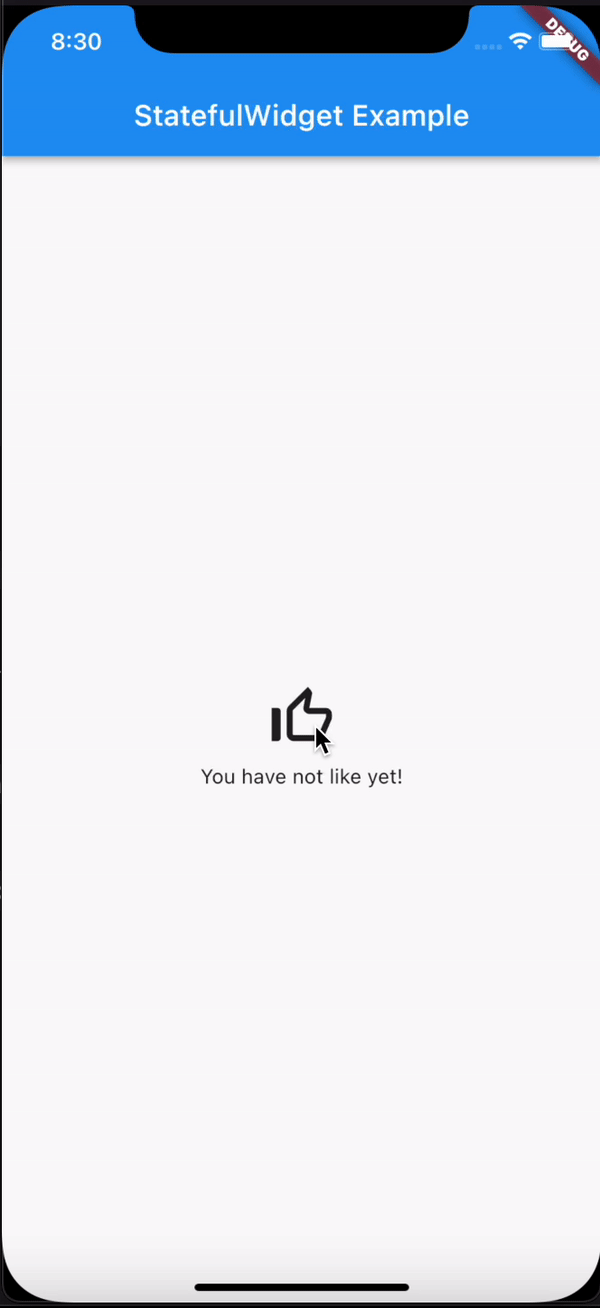
}

}

***Giải thích:***

1. Khai báo biến kiểu String trong State object của MyHomePage. Đây sẽ là nơi nhận sự thay đổi text.
2. Lúc này Text sẽ sử dụng biến state chúng ta vừa tạo thay vì text set cứng như lúc đầu.
3. Truyền vào đối số cho callback function của LikeButton.
4. Khi nhận được callback trả về state của LikeButton thì MyHomePage sẽ gọi hàm setState() để set text tương ứng cho likedStatusText từ đó kích hoạt Flutter render lại UI.

#### Kết quả



#### [Demo source code](https://github.com/vanle57/stateless-stateful-widget/tree/main/demo%20source%20code?ref=200lab.io)

## 7. Tổng kết

Tóm lại sự khác nhau rõ rệt và có thể dễ dàng thấy được nhất giữa StalessWidget và StatefulWidget chính là khả năng reload của widget ở runtime. Nếu như StalessWidget không thể thay đổi chỉ trừ khi được khởi tạo lại thì StatefulWidget có các trạng thái có thể thay đổi được và Flutter sẽ render lại UI cho phù hợp với từng trạng thái.

## [Các thư viện state managerment cho Flutter](https://blog.vietnamlab.vn/cac-thu-vien-state-cho-flutter/)

Flutter là một framework phát triển ứng dụng di động và web đa nền tảng được phát triển bởi Google. Nó cho phép bạn xây dựng ứng dụng đẹp và tương tác bằng cách sử dụng một ngôn ngữ duy nhất, Dart. Điểm đặc biệt của Flutter là khả năng tạo giao diện người dùng đồ họa nhanh chóng và đồng nhất trên nhiều nền tảng, giúp tiết kiệm thời gian và công sức trong phát triển ứng dụng đa nền tảng.

State management là một khía cạnh quan trọng trong phát triển ứng dụng Flutter. Vì Flutter thường xây dựng các giao diện người dùng động và tương tác, nên việc quản lý trạng thái (state) của ứng dụng là cực kỳ quan trọng. State management giúp bạn theo dõi và điều khiển dữ liệu và trạng thái của ứng dụng một cách hiệu quả. Nó còn giúp tăng tính linh hoạt và bảo trì mã nguồn dễ dàng hơn khi ứng dụng phức tạp hóa.

Có nhiều cách để quản lý trạng thái trong Flutter, bao gồm: [**StatefulWidget**](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html)**,**[**Provider**](https://pub.dev/packages/provider)**,**[**Bloc**](https://bloclibrary.dev/#/)**,**[**GetX**](https://pub.dev/packages/get)**,...**  
Sau đây chúng ta sẽ đi vào chi tiết từng thư viện, điểm mạnh, điểm yếu và cách triển khai đơn giản nhất của mỗi thư viện. Nào let's go...!!

### I. ****Provider:****

**1. Điểm mạnh:**

* Dễ dàng để bắt đầu và sử dụng.
* Hiệu suất tốt và tương thích với các tính năng mới của Flutter.
* Cộng đồng lớn và hỗ trợ tài liệu phong phú.

**2. Điểm yếu:**

* Có thể trở nên phức tạp với các ứng dụng lớn.
* Không cung cấp sự quản lý trạng thái toàn cục mặc định.

**3. Demo code:**

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: ChangeNotifierProvider(

create: (context) => MyState(),

child: MyHomePage(),

),

);

}

}

class MyState with ChangeNotifier {

int count = 0;

void increment() {

count++;

notifyListeners();

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

final myState = Provider.of<MyState>(context);

return Scaffold(

appBar: AppBar(

title: Text('Provider Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Count: ${myState.count}'),

RaisedButton(

onPressed: () {

myState.increment();

},

child: Text('Increment'),

),

],

),

),

);

}

}

Provider thích hợp cho các dự án Flutter đơn giản hoặc có quy mô trung bình. Nó làm tốt trong hầu hết các trường hợp và rất dễ để bắt đầu.

### II. ****Riverpod:****

**1. Điểm mạnh:**

* Mô hình quản lý trạng thái mạnh mẽ và an toàn.
* Hỗ trợ tính năng phụ thuộc động linh hoạt.
* Dễ dàng kiểm tra và test.

**2. Điểm yếu:**

* Tích hợp với Flutter không phải lúc nào cũng trực quan.
* Tài liệu và cộng đồng chưa phát triển như Provider.

**3. Demo code:**

import 'package:flutter/material.dart';

import 'package:riverpod/riverpod.dart';

void main() {

runApp(MyApp());

}

final countProvider = StateProvider<int>((ref) => 0);

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyHomePage(),

);

}

}

class MyHomePage extends ConsumerWidget {

@override

Widget build(BuildContext context, ScopedReader watch) {

final count = watch(countProvider);

return Scaffold(

appBar: AppBar(

title: Text('Riverpod Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Count: ${count.state}'),

RaisedButton(

onPressed: () {

count.state++;

},

child: Text('Increment'),

),

],

),

),

);

}

}

Riverpod được đề xuất cho các dự án có độ phức tạp cao hơn và cần kiểm soát trạng thái mạnh mẽ hơn. Nó thích hợp cho các ứng dụng lớn và phức tạp hơn.

### III. ****GetX:****

**1. Điểm mạnh:**

* Thư viện đa chức năng với quản lý trạng thái, định tuyến và nhiều tính năng khác.
* Hiệu suất cao và cú pháp đơn giản.
* Hệ thống định tuyến tích hợp và quản lý trạng thái toàn cục.

**2. Điểm yếu:**

* Cách tiếp cận đôi khi có thể làm cho mã nguồn trở nên phức tạp với các ứng dụng lớn.
* Cộng đồng và tài liệu không phát triển bằng các thư viện khác.

**3. Demo code:**

import 'package:flutter/material.dart';

import 'package:get/get.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: MyHomePage(),

);

}

}

class MyHomePage extends StatelessWidget {

final count = 0.obs;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('GetX Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Count: $count'),

RaisedButton(

onPressed: () {

count.value++;

},

child: Text('Increment'),

),

],

),

),

);

}

}

GetX là lựa chọn tốt cho các dự án cần tích hợp quản lý trạng thái, định tuyến và giao diện người dùng trong một gói duy nhất. Nó đơn giản và hiệu quả cho các ứng dụng vừa và nhỏ.

### IV. ****Bloc:****

**1. Điểm mạnh:**

* Tách biệt hoàn toàn logic ứng dụng và giao diện người dùng.
* Dễ dàng quản lý trạng thái và xử lý tương tác người dùng.
* Khả năng kiểm tra và tái sử dụng mã nguồn dễ dàng.

**2. Điểm yếu:**

* Yêu cầu hiểu biết sâu về kiến trúc và sự cách ly.
* Cú pháp đôi khi có thể trở nên phức tạp.

**3. Demo code:**

import 'package:flutter/material.dart';

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

void main() {

runApp(MyApp());

}

enum CounterEvent { increment, decrement }

class CounterBloc extends Bloc<CounterEvent, int> {

CounterBloc() : super(0);

@override

Stream<int> mapEventToState(CounterEvent event) async\* {

switch (event) {

case CounterEvent.increment:

yield state + 1;

break;

case CounterEvent.decrement:

yield state - 1;

break;

}

}

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

home: BlocProvider(

create: (context) => CounterBloc(),

child: MyHomePage(),

),

);

}

}

class MyHomePage extends StatelessWidget {

@override

Widget build(BuildContext context) {

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

return Scaffold(

appBar: AppBar(

title: Text('Bloc Example'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

BlocBuilder<CounterBloc, int>(

builder: (context, state) {

return Text('Count: $state');

},

),

RaisedButton(

onPressed: () {

counterBloc.add(CounterEvent.increment);

},

child: Text('Increment'),

),

RaisedButton(

onPressed: () {

counterBloc.add(CounterEvent.decrement);

},

child: Text('Decrement'),

),

],

),

),

);

}

}

Bloc là một lựa chọn tốt cho các dự án đòi hỏi kiến trúc rõ ràng và cách ly giữa logic ứng dụng và giao diện người dùng. Nó phù hợp cho các ứng dụng có tính phức tạp và yêu cầu kiểm tra và tái sử dụng cao.

### V. Tổng kết:

Tóm lại, lựa chọn thư viện quản lý trạng thái trong Flutter nên dựa trên độ phức tạp của dự án, trình độ kỹ thuật của bạn và tính năng cụ thể mà bạn cần. Provider là lựa chọn tốt cho các dự án đơn giản, trong khi Riverpod và GetX phù hợp cho các dự án phức tạp hơn. Bloc là lựa chọn cho các ứng dụng đòi hỏi kiến trúc rõ ràng và sự cách ly. Mọi thư viện đều có điểm mạnh và điểm yếu riêng, vì vậy nên cân nhắc kỹ trước khi quyết định.

# **CÁC CÁCH QUẢN LÝ STATE TRONG FLUTTER – Provider Basic**

Provider trong Flutter là một trong những kiến thức cơ bản mà bất cứ lập trình viên Flutter nào cần phải được nắm vững.

Provider trong lập trình Flutter được coi như là một bản nâng cấp so với cách quản lý trạng thái : [InheritedWidget](https://baoflutter.com/nghe-thuat-flutter-cac-cach-quan-ly-state-trong-flutter-inheritedwidget/) mà tôi đã có bài viết trong chuyên đề nghệ thuật Flutter.

Provider được xây dựng cũng theo tư tưởng như InheritedWidget nhưng dễ sử dụng và rất nhiều tính năng ưu viêt như:  
+ Phân bổ và xử lý tài nguyên đơn giản  
+ lazy loading  
+ Dễ dùng và thân thiện trong những project lớn và phức tạp

Ngoài ra, provider còn được tích hợp vào trong các cách quản lý trạng thái (State Management) khác như : flutter\_bloc, Mobx

**Nguyên lý**  
Provider được tạo ra dựa trên InheritedWidget, vì vậy nguyên lý của nó cũng tương tự,

Khi khai báo provider trên một Widget tổ tiên( cha, ông.. )-ancestor widget thì các Widget con cháu có thể sử dụng và cập nhật giá trị. Khi giá trị của model đăng kí provider thay đổi thì tất cả các Widget con cháu nào mà liên quan đến các thuộc tính của phần tử này sẽ được cập nhật lại.

Dưới đây tôi sẽ đưa ra một ví dụ: Tạo ra các widgets là các box và đặt đăng kí provider ở trước Widget cha. Các Widget con, có box sử dụng thuộc tính provider, có box thì không. Khi thay đổi giá trị các thuộc tính thì tất cả các Widget liên quan sẽ thay đổi.



Bắt đầu nào !

Tạo một Flutter project mới:

**Cài đặt thư viện**

**Khai báo thư viện trên pubspec.yaml : provider**  
Ví dụ : Nhấn Pub get hoặc run trên Terminal : flutter packages get

provider 4.3.2+2

**Tạo một lớp Feature**  
Lớp này chứa các thuộc tính như : count, sử dụng ChangeNotifier để lắng nghe những sự thay đổi.

Class này chứa các phương thức và thuộc tính, đặt lắng nghe khi có sự thay đổi bằng việc sử dụng ChangeNotifier

//

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

class Feature with ChangeNotifier {

int \_count = 0;

int get count => \_count;

/// Tăng số

increaseCount()

{

\_count ++;

notifyListeners(); // khi thay đổi sẽ lắng nghe và thông báo đến các Widget liên quan

}

}

**2. Tạo các Widgets, khai báo provider và sử dụng các thuộc tính, phương thức của model trên.**

\* Khai báo Provider

Khai báo provider được thực hiện trước các Widget là Widget tổ tiên của các Widget con mà sử dụng các thuộc tính, phương thức của model mà đăng kí provider.  
Thông thường đăng kí provider ở hàm main hay trong class đầu tiên của ứng dụng. Ví dụ :

void main(){

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MultiProvider(

providers: [

ChangeNotifierProvider(create: (context) => Feature()),

],

child:MaterialApp(

home: Scaffold(

body: Center(

child: Container(

child: Box1(),

),

),

),

));

}

}

Từ đoạn code trên chúng ta biết được cách đăng kí Provider :

void main(){

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MultiProvider(

providers: [

ChangeNotifierProvider(create: (context) => Feature()),

],

child:MaterialApp(

home: Scaffold(

body: Center(

child: Container(

child: Box1(),

),

),

),

));

}

}

\* Tiếp theo là tạo các widget Box1, và Box21, Box22, Box23 là các Widget con của Box1 , các Widget : Box21, Box23 sẽ sử dụng thuộc tính của model : Feature thông qua provider còn Box22 thì không.

Do vậy khi thay đổi giá trị count ở Box1 thì Widget Box21, Box23 sẽ thay đổi

Các sử dụng thuộc tính:  
\* Dùng Watch khi lắng nghe để hiển thị

...

context.watch<Feature>().count

\* Dùng read khi muốn dùng để thay đổi

// Tăng số

context.read<Feature>().increaseCount();

// Thay đổi giá trị count ở Box1

context.read<Feature>().count = 5;

**Box1**

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'package:provider\_base/feature.dart';

import 'box21.dart';

import 'box22.dart';

import 'box23.dart';

class Box1 extends StatefulWidget {

@override

\_Box1State createState() => \_Box1State();

}

class \_Box1State extends State<Box1> {

@override

Widget build(BuildContext context) {

return Container(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Box21(),

Box22(),

Box23(),

increaseNumber()

],

),

);

}

// Button of increasing Number

increaseNumber()

{

return RaisedButton(

child: Text("Tăng số", style: TextStyle(fontSize: 20),),

onPressed: (){

context.read<Feature>().increaseCount();

},

);

}

}

**Box21**

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

import 'package:provider\_base/feature.dart';

import 'box21.dart';

import 'box22.dart';

import 'box23.dart';

class Box1 extends StatefulWidget {

@override

\_Box1State createState() => \_Box1State();

}

class \_Box1State extends State<Box1> {

@override

Widget build(BuildContext context) {

return Container(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Box21(),

Box22(),

Box23(),

increaseNumber()

],

),

);

}

// Button of increasing Number

increaseNumber()

{

return RaisedButton(

child: Text("Tăng số", style: TextStyle(fontSize: 20),),

onPressed: (){

context.read<Feature>().increaseCount();

},

);

}

}**Box22**

import 'package:flutter/material.dart';

import 'package:provider\_base/feature.dart';

import 'package:provider/provider.dart';

class Box22 extends StatefulWidget {

@override

\_Box22State createState() => \_Box22State();

}

class \_Box22State extends State<Box22> {

@override

Widget build(BuildContext context) {

return Container(

height: 100,

width: 200,

margin : EdgeInsets.only(top: 10),

decoration: BoxDecoration(

color: Colors.grey ,

),

child: Center(

child: Text("Không dùng Provider", style: TextStyle(color: Colors.white, fontSize: 20),),

)

);

}

}

**Box23**

import 'package:flutter/foundation.dart';

import 'package:flutter/material.dart';

class Feature with ChangeNotifier {

  int \_count = 0;

  int get count => \_count;

  /// Tăng số

  increaseCount()

  {

    \_count ++;

    notifyListeners(); // khi thay đổi sẽ lắng nghe và thông báo đến các Widget liên quan

  }

}

Kết quả thì như hình bên trên.  
Sourcecode của bài viết này trên [Github](https://github.com/PhamBao91/baoflutter/tree/master/provider_base).

Như vậy, các bạn đã biết các sử dụng provider một các cơ bản. Tóm lại bao gồm các bước sau:  
+ Khai báo thư viện  
+ Tạo model và đặt lắng nghe với ChangeNotifier  
+ Khai báo provider ở các Widget cha  
+ Sử dụng provider ở các Widget con.