


Flutter Journal

 [Common Widgets](#)

 [Dart](#)

Cài đặt Flutter

Kiến thức

Cài đặt Flutter

cmd/PowerShell

Android Studio

Sử dụng điện thoại để hiển thị phần mềm

Terminal

Thêm thư viện

<https://pub.dev/>

Setup firebase

follow step 3 in

Add Firebase SDK

Notes

- [Cài biến môi trường](#) (PATH)

```
flutter doctor (kiểm tra các phần mềm còn thiếu)
flutter create ... (Tạo lập dự án)
```

- Tạo máy ảo
- Tạo dự án thử nghiệm
- Cắm cổng Type-C
- Cài đặt điện thoại ở chế độ nhà phát triển

```
flutter clean
flutter pub get
Run
```

```
flutter pub add {name}
```

```
android {
  defaultConfig {
    // ...
    minSdkVersion 17
    targetSdkVersion 30
    multiDexEnabled true
  }
}
```

add google-services.json
file in android/app folder

```
dependencies {  
    implementation 'com.android.support.multidex:1.0.3'  
}
```

```
classpath 'com.android.tools.build:gradle:7.3.0'  
classpath 'com.google.gms:google-services:4.4.0'  
//instead of id 'com.google.gms.google-services' version '4.4.0' ap
```

Khái niệm thẻ-Widget



SUMMARY: widget là mọi thứ trong Flutter (widgets are everything in Flutter framework). Đây là thành phần cơ bản và chủ yếu nhất. Nó tương tự với các **element** ở trong **HTML**

Các nhóm widgets

Platform widgets

Layout widgets

State maintenance widgets

Platform independent / basic
widgets

Notes

- Các widget giao diện đặc thù theo từng nền tảng
- Các widget hỗ trợ bố trí giao diện
- Các widget quản lý trạng thái
- Các widget cơ bản độc lập với nền tảng

Cách tạo thẻ-Widget

```
//You can define your own Widget  
class MyApp extends StatelessWidget {  
    //Let's define a constructor here  
    String name;  
    int age;  
    MyApp({this.name, this.age});  
    //StatelessWidget: widget which donot have "state"  
    //(property which has been changed <-> User Interface)  
    @override  
    Widget build(BuildContext context) {  
        // build: abstract method from parent class(StatelessWidget)  
        return MaterialApp(  
            title: "This is my first app",  
            home: Scaffold(  
                body: Center(child: Text(  
                    "Mr Hoang, name: $name, age : $age",  
                    //How to increase TextSize ?  
                    style: TextStyle(  
                        fontSize: 30,  
                        fontWeight: FontWeight.bold,  
                        color: Colors.red  
                    ),  
                    textDirection: TextDirection.ltr//left-to-right  
                )), ));  
    }  
}
```

Gesture trong Flutter

Một số cử chỉ được sử dụng rộng rãi:

- **Tap** – Chạm vào bề mặt thiết bị bằng đầu ngón tay trong thời gian ngắn sau đó thả ngón tay ra ngay

- **Double Tap** – Tap 2 lần trong thời gian ngắn
- **Drag** – Chạm vào bề mặt của thiết bị bằng đầu ngón tay và sau đó di chuyển đầu ngón tay một cách ổn định và cuối cùng thả ngón tay ra.
- **Flick** – Tương tự như drag nhưng thực hiện nhanh hơn.
- **Pinch** – Chạm bề mặt của thiết bị bằng hai ngón tay
- **Spread/Zoom** – Ngược lại với **Pinch**.
- **Panning** – Chạm vào bề mặt của thiết bị bằng đầu ngón tay và di chuyển nó theo bất kỳ hướng nào mà không nhả đầu ngón tay.



Để xác định các cử chỉ tác động lên một **widget**, ta chỉ cần đặt **widget** đó bên trong **GestureDetector** widget. **GestureDetector** sẽ bắt các cử chỉ và gửi nhiều sự kiện dựa trên cử chỉ đó.

Một số cử chỉ và các sự kiện tương ứng

DateTap	Double tap	Long press	Vertical drag	Horizontal drag	Pan
onTapDown	onDoubleTap	onLongPress	onVerticalDragStart	onHorizontalDragStart	onPanStart
onTapUp			onVerticalDragUpdate	onHorizontalDragUpdate	onPanUpdate
onTap			onVerticalDragEnd	onHorizontalDragEnd	onPanEnd
onTapCancel					

```
body: Center(
  child: GestureDetector(
    onTap: () {
      _showDialog(context);
    },
    child: Text( 'Hello World', )
  ),
),
```

```
// user defined function void _showDialog(BuildContext)
// flutter defined function
showDialog(
  context: context, builder: (BuildContext con
  // return object of type Dialog
  return AlertDialog(
    title: new Text("Message"),
    content: new Text("Hello World"),
    actions: <Widget>[
      new FlatButton(
        child: new Text("Close"),
        onPressed: () {
          Navigator.of(context).pop();
        },
      ),
    ],
  );
}
```

▼ Flutter còn cung cấp một số nhỏ các widget để thực hiện các cử chỉ cụ thể đơn giản cũng như phức tạp:

- **Dismissible** – Hỗ trợ flick gesture để đóng widget.
- **Draggable** – Hỗ trợ drag gesture để di chuyển widget.
- **LongPressDraggable** – Hỗ trợ drag gesture để duy chuyển widget, khi widget cha có thể kéo thả
- **DragTarget** – Chấp nhận *Draggable* widget
- **IgnorePointer** – Ẩn widget
- **AbsorbPointer** – Dừng việc xử lý cử chỉ trên widget
- **Scrollable** – Hỗ trợ cuộn nội dung trong widget

StatefulWidget vs StatelessWidget

SS

Bản chất

StatelessWidget

Là widget không có trạng thái nội bộ.

StatefulWidget

Là widget có trạng thái nội bộ.

Dữ liệu của nó có thể thay đổi trong quá trình thực thi.

Dữ liệu của nó không thay đổi trong quá trình thực thi.

Dữ liệu thay đổi

Không thể thay đổi trực tiếp dữ liệu của `StatelessWidget` sau khi đã được tạo.

Có một lớp con tương ứng với `StatefulWidget` gọi là `State`.

`build` method

`build` method được gọi mỗi khi cần vẽ lại widget.
Nó nhận giá trị từ các thuộc tính đã được khai báo và sử dụng chúng để xây dựng giao diện.

Dữ liệu có thể thay đổi trong `State` và khi có sự thay đổi, `build` method của `StatefulWidget` được gọi lại.

`build` method được gọi lại mỗi khi có sự thay đổi trong trạng thái.



`StatelessWidget` thường được sử dụng cho các phần của giao diện không thay đổi.

`StatefulWidget` thường được sử dụng khi giao diện cần phản ánh sự thay đổi trong trạng thái hoặc dữ liệu nội bộ.

ScopedModel trong Flutter



Scope_model - chúng ta có thể hiểu đơn giản nó là một framework, ví dụ khi bạn code "thô", lúc data model thay đổi, ở hàm `setState()` thay vì rebuild lại tất cả các widget (Btn, txt, ..) thì sẽ phạm phải quy tắc Single Responsibility thì `Scope_model` được sinh ra với mục đích chỉ rebuild data model bị thay đổi thay vì rebuild tất cả widget

```
class Product extends Model {
  final String name;
  final String description;
  final int price;
  final String image;
  int rating;

  Product(this.name, this.description, this.price, this.image, this.rating);
  factory Product.fromMap(Map<String, dynamic> json) {
    return Product(
      json['name'],
      json['description'],
      json['price'],
      json['image'],
      json['rating'],
    );
  }
  void updateRating(int myRating) {
    rating = myRating; notifyListeners();
  }
}
```

Animation

Trong flutter, hệ thống animation không có bất kì animation cụ thể nào cả. Thay vào đó nó cung cấp duy nhất giá trị yêu cầu cho tất cả các frame để render hình ảnh

Animation

Các kiểu animation thông dụng

Notes

- `Animation<double>`: Thêm các giá trị giữa hai số thập phân

- *Animation<Color>* : Thêm các màu vào giữa hai màu
- *Animation<Size>* : Thêm kích thước vào giữa hai kích thước
- *AnimationController* : Là một đối tượng animation đặc biệt dùng để điều khiển các hiệu ứng của chính nó.

```
controller = AnimationController(duration: const Duration(seconds: 2), vsync: this);
```

Nó cơ bản giống như *AnimationController* nhưng hỗ trợ animation phi tuyến tính (đường cong).

CurvedAnimation

```
controller = AnimationController(duration: const Duration(seconds: 2), vsync: this);
animation = CurvedAnimation(parent: controller, curve: Curves.easeInOut);
```

Được kế thừa từ *Animatable<T>* và tạo các giá trị bất kì khác 0 và 1. Nó được sử dụng cùng với đối tượng animation bởi phương thức *animate*

Tween<T>

```
AnimationController controller = AnimationController(
  duration: const Duration(milliseconds: 1000), vsync: this);
Animation<int> customTween = IntTween(begin: 0, end: 255).animate(controller);
```

Ngoài ra, Tween cũng có thể sử dụng cùng với *CurvedAnimation*

```
AnimationController controller = AnimationController(
  duration: const Duration(milliseconds: 500), vsync: this);
final Animation curve = CurvedAnimation(parent: controller, curve: Curves.easeInOut);
Animation<int> customTween = IntTween(begin: 0, end: 255).animate(controller);
```

Cách tạo Animation

```
class MyApp extends StatefulWidget {
  _MyAppState createState() => _MyAppState();
}
```

```
class _MyAppState extends State<MyApp> with SingleTickerProviderStateMixin {
  Animation<double> animation;
  AnimationController controller;
  @override void initState() {
    super.initState();
    controller = AnimationController(
      duration: const Duration(seconds: 10), vsync: this
    );
    animation = Tween<double>(begin: 0.0, end: 1.0).animate(controller);
    controller.forward();
  }
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    controller.forward();
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(primarySwatch: Colors.blue,),
      home: MyHomePage(title: 'Product layout demo home page', animation: animation,)
```

```

    );}
    @override
    void dispose() {
        controller.dispose();
        super.dispose();
    }
}

```

Hàm initState được dùng để tạo đối tượng animation controller (controller), đối tượng animation(animation) và để bắt đầu với animation, chúng ta sử dụng phương thức controller.forward.

Ở hàm dispose, như mình nói ở trên, sau khi tạo thì chúng ta phải hủy nó nên đó là chức năng của hàm dispose để hủy bỏ controller

Ở hàm Build, animation được gửi tới MyHomePage widget thông qua constructor. Bây giờ, MyHomePage có thể sử dụng đối tượng animation tạo hiệu ứng cho nội dung

```

class MyAnimatedWidget extends StatelessWidget {
    MyAnimatedWidget({this.child, this.animation});
    final Widget child;
    final Animation<double> animation;
    Widget build(BuildContext context) => Center(
        child: AnimatedBuilder(
            animation: animation,
            builder: (context, child) => Container(
                child: Opacity(opacity: animation.value, child: child),
            ),
            child: child),
    ); }

```

```

class MyHomePage extends StatelessWidget {
    MyHomePage({super.key, this.title, this.animation});
    final String title;
    final Animation<double>
    animation;
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(title: Text("Product Listing")),body: ListView(
                shrinkWrap: true,
                padding: const EdgeInsets.fromLTRB(2.0, 10.0, 2.0, 10.0),
                children: <Widget>[
                    FadeTransition(
                        child: ProductBox(
                            name: "iPhone",
                            description: "iPhone is the stylist phone ever",
                            price: 1000,
                            image: "iphone.jpg"
                        ), opacity: animation
                    ),
                    MyAnimatedWidget(child: ProductBox(
                        name: "Pixel",
                        description: "Pixel is the most featureful phone ever",
                        price: 800,
                        image: "pixel.jpg"
                    ), animation: animation),
                ],
            ),
        );
    }
}

```

Code với native Android

viết ứng dụng đơn giản để mở trình duyệt web sử dụng Android SDK

```
package com.tutorialspoint.flutterapp.flutter_browser_app;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import io.flutter.app.FlutterActivity;
import io.flutter.plugin.common.MethodCall;
import io.flutter.plugin.common.MethodChannel.Result;
import io.flutter.plugins.GeneratedPluginRegistrant;


public class MainActivity extends FlutterActivity {
  ///đặt tên Channel
  private static final String CHANNEL = "flutterapp.tutorialspoint.com/browser";
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    GeneratedPluginRegistrant.registerWith(this);
    ///viết mã cụ thể cho Android để xử lý message
    new MethodChannel(getFlutterView(), CHANNEL).setMethodCallHandler(
      new MethodCallHandler() {
        @Override
        public void onMethodCall(MethodCall call, Result result) {
          String url = call.argument("url");
          if (call.method.equals("openBrowser")) {
            openBrowser(call, result, url);
          } else {
            result.notImplemented();
          }
        }
      }
    );
    ///viết hàm openBrowser để mở trình duyệt
    private void openBrowser(MethodCall call, Result result, String url) {
      Activity activity = this; if (activity == null) {
        result.error(
          "ACTIVITY_NOT_AVAILABLE", "Browser cannot be opened without foreground activity",
          );
        return;
      }
      Intent intent = new Intent(Intent.ACTION_VIEW);
      intent.setData(Uri.parse(url));
      activity.startActivity(intent);
      result.success((Object) true);
    }
  }
}
```

```
import 'package:flutter/material.dart';
import 'dart:async';
import 'package:flutter/services.dart';
class MyHomePage extends StatelessWidget {
  MyHomePage({super.key, this.title});
  final String title;
  static const platform = const MethodChannel('flutterapp.tutorialspoint.com/browser');
  ///viết phương thức _openBrowser để gọi nền tảng cụ thể thông qua message channel.
  Future<void> _openBrowser() async {
    try {
      final int result = await platform.invokeMethod('openBrowser', <String, String>{
        'url': "https://flutter.dev"
      });
    }
  }
}
```

```


    });}
    on PlatformException catch (e) {
      // Unable to open the browser print(e);
    }
  }
  ///tạo một nút có chức năng để mở trình duyệt
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(this.title),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Open Browser'),
          onPressed: _openBrowser,
        ),
      ),
    );
  }
}

```

 **SUMMARY:** Nhìn trên, chúng ta đã tạo *message channel* sử dụng lớp **MethodChannel** và lớp **MethodCallHandler** để xử lý thông báo. **onMethodCall** có trách nhiệm gọi đúng mã nền tảng riêng biệt bằng cách kiểm tra thông báo. Hàm **onMethodCall** đọc url từ thông báo và gọi đến `openBrowser` khi mà hàm gọi `openBrowser`. Ngược lại hàm sẽ trả về method *notImplemented*

Database

SQLite

 **SQLite** là một SQL tiêu chuẩn dựa trên công cụ cơ sở dữ liệu nhúng. Gói **sqflite** cung cấp nhiều chức năng để làm việc hiệu quả với **SQLite** database.

Database


Notes

SQLite

Chức năng chính

- **SQLite** là một SQL tiêu chuẩn dựa trên công cụ cơ sở dữ liệu nhúng. Gói **sqflite** cung cấp nhiều chức năng để làm việc hiệu quả với **SQLite** database.
- Tạo/mở SQLite database
- Thực thi SQL statement (thực thi phương thức) đối với SQLite database
- Phương thức truy vấn nâng cao (phương thức truy vấn) để giảm code cần thiết để truy vấn và lấy thông tin từ SQLite database.

Firestore

 **SQLite** là một SQL tiêu chuẩn dựa trên công cụ cơ sở dữ liệu nhúng. Gói **sqflite** cung cấp nhiều chức năng để làm việc hiệu quả với **SQLite** database.

```

//cai dat firebase-tools
npm install -g firebase-tools
//kiem tra tai khoan firebase da login dung chua
firebase login
//kich hoat flutterfire_cli
dart pub global activate flutterfire_cli
//chon project muon connect
flutterfire configure

```



```
flutter pub add firebase_core
flutter pub add firebase_messaging
```

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);
  runApp(const MyApp());
}
```

Chuyển đổi ngôn ngữ

Các lớp cơ sở

Locale

```
Locale en_locale = Locale('en', 'US')
```

Localizations

Notes

Là lớp được sử dụng để nhận diện ngôn ngữ người sử dụng.

Đối số đầu tiên là mã ngôn ngữ, đối số thứ hai là mã quốc gia.

là widget chung được sử dụng để set Locale và nguồn localized của lớp con

Cách tạo ứng dụng đa ngôn ngữ cơ bản

```
flutter pub add flutter_localization
flutter pub add intl
flutter pub add intl_translation
```

```
void main() {
  runApp(const MyLocalization());
}
class MyLocalization extends StatelessWidget {
  const MyLocalization({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: MyHomePage(),
    );
  }
}
//Thêm quyền cho MaterialApp, WidgetsApp và CustomLocalization
//sử dụng thuộc tính MaterialApp, localizationsDelegates và supportedLocales
localizationsDelegates: const [
  CustomLocalizationsDelegate(),
  GlobalMaterialLocalizations.delegate,
  GlobalWidgetsLocalizations.delegate,
],
supportedLocales: const [
  Locale('en', ''),
  Locale('es', ''),
],);}}
class MyHomePage extends StatelessWidget {
  const MyHomePage({super.key});
  @override
  Widget build(BuildContext context) {
```

```

return Scaffold(
  appBar: AppBar(title: Text(CustomLocalizations.of(context).title),),
  body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Text(CustomLocalizations.of(context).message,),
      ],),),),);}}

```

Cách tạo ứng dụng đa ngôn ngữ cơ bản

```

import 'package:flutter/foundation.dart' show SynchronousFuture;
class CustomLocalizations {
  CustomLocalizations(this.locale);
  final Locale locale;
  static CustomLocalizations of(BuildContext context) {
    return Localizations.of<CustomLocalizations>(context, CustomLocalizations)!;
  }
  static Map<String, Map<String, String>> _localizedValues = {
    'en': {
      'title': 'Demo',
      'message': 'Hello World',
    },
    'es': {
      'title': 'Manifestación',
      'message': 'Hola Mundo',
    },
  };
  String get title {
    return _localizedValues[locale.languageCode]!['title']!;
  }
  String get message {
    return _localizedValues[locale.languageCode]!['message']!;
  }
}
class CustomLocalizationsDelegate extends LocalizationsDelegate<CustomLocalizations> {
  const CustomLocalizationsDelegate();
  @override
  bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);
  @override
  Future<CustomLocalizations> load(Locale locale) {
    return SynchronousFuture<CustomLocalizations>(CustomLocalizations(locale));
  }
  @override
  bool shouldReload(CustomLocalizationsDelegate old) => false;
}

```

Sử dụng gói intl

```

import 'package:intl/intl.dart';
import 'package:internationalization/l10n/messages_all.dart';
class CustomLocalizations {
  static Future<CustomLocalizations> load(Locale locale) {
    final String name = (locale.countryCode?.isEmpty ?? true)
      ? locale.languageCode : locale.toString();
    final String localeName = Intl.canonicalizedLocale(name);
    return initializeMessages(localeName).then((_) {
      Intl.defaultLocale = localeName;
      return CustomLocalizations();
    });
  }
}

```

```

});}
static CustomLocalizations of(BuildContext context) {
  return Localizations.of<CustomLocalizations>(context, CustomLocalizations)!;
}
String get title {
  return Intl.message(
    'Demo',
    name: 'title',
    desc: 'Title for the Demo application',
  );}
String get message {
  return Intl.message(
    'Hello World',
    name: 'message',
    desc: 'Message for the Demo application',
  );}
}
class CustomLocalizationsDelegate extends LocalizationsDelegate<CustomLocalizations> {
  const CustomLocalizationsDelegate();
  @override
  bool isSupported(Locale locale) => ['en', 'es'].contains(locale.languageCode);
  @override
  Future<CustomLocalizations> load(Locale locale) {
    return CustomLocalizations.load(locale);
  }
  @override
  bool shouldReload(CustomLocalizationsDelegate old) => false;
}

```

1. Intl.canonicalizedLocale - được sử dụng để lấy chính xác tên ngôn ngữ
2. Intl.defaultLocale - Sử dụng để set ngôn ngữ hiện tại
3. Intl.message - Sử dụng để định nghĩa thông điệp mới

```

tạo thư mục lib/l10n
import 'l10n/messages_all.dart';

```

```
flutter packages pub run intl_translation:extract_to_arb --output-dir=lib/l10n lib/main.dart
```

```
flutter packages pub run intl_translation:generate_from_arb --output-dir=lib\l10n --no-use-de
```

Xuất ứng dụng trong Flutter

```

//////Cách xuất ứng dụng sang file APK
flutter build apk
//////Cách cài đặt file APK trực tiếp trên thiết bị di động
flutter install
//////Đẩy ứng dụng lên Google PlayStore bằng cách tạo appbundle và đẩy nó lên
flutter build appbundle

```

```

//////Cách xuất ứng dụng sang file APK
flutter build ios

```

State Management in Flutter

SetState

Notes

setState()



Ưu điểm:

Đơn giản và Dễ Hiểu

Phù Hợp Cho Trạng Thái Nhỏ và Đơn Giản

Tự Động Gọi lại `build`

Nhược điểm

Không Phát Hiện Sự Thay Đổi

Áp Dụng Chỉ Cho Widget Hiện Tại và Con Của Nó

Không Phù Hợp Cho Ứng Dụng Lớn và Phức Tạp

Dùng để chuyển dữ liệu từ cha sang con

```
int counterOngBa = 0;
class OngBa extends StatefulWidget {
  const OngBa({super.key});
  @override
  State<OngBa> createState() => _OngBaState();
}
class _OngBaState extends State<OngBa> {
  @override
  Widget build(BuildContext context) {
    counterOngBa++;
    print('Class OngBa build lan thu $counterOngBa');
    return Directionality(
      textDirection: TextDirection.ltr,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Center(
            child: ElevatedButton(
              onPressed: () {
                setState(() {
                  print('Class OngBa setState lan thu $count
                ));},
            child: Text('OngBa build lan thu $counterOngBa
          ),
        BoMe(), //truyen du lieu sang class BoMe()
        CoChu(), //truyen du lieu sang class CoChu()
      ],),),);}}
```

Provider

Sử dụng thư viện Provider

Notes

```
flutter pub add provider
```

```
import 'package:provider/provider.dart';
```

```
void main() {
  runApp(ChangeNotifierProvider(
    create: (_) => CounterProvider(),
    child: const MaterialApp(home: MyProvider())
  ));
}
```

```
class CounterProvider extends ChangeNotifier {
  int _counter = 100;
  int get counter => _counter;
  void add() {
    _counter++;
    notifyListeners();
  }
}
```

```
class MyProvider extends StatelessWidget {
  const MyProvider({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
```



```
class MyProvider extends StatelessWidget {
  const MyProvider({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Home Screen'),
        backgroundColor: Colors.greenAccent,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
```

```
        title: const Text('Home Screen'),
        backgroundColor: Colors.greenAccent,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              ///Nhưng cach khac
              // context.watch<CounterProvider>().counter
              // Provider.of<CounterProvider>(context, li
              Provider.of<CounterProvider>(context).count
              style: const TextStyle(fontSize: 50),
            ),
            ElevatedButton(
              onPressed: () {
                Navigator.push(context,
                  MaterialPageRoute(builder: (context) => c
                ),
                child: const Text('Go to second Screen'))
          ],)),
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.add),
        onPressed: () {
          context.read<CounterProvider>().add();
          ///cach khac
          ///Provider.of<CounterProvider>(context, li
        },),,));}}
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Screen'),
        backgroundColor: Colors.amberAccent,
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              ///Nhưng cach khac
              // context.watch<CounterProvider>().counte
              // Provider.of<CounterProvider>(context, 1
              Provider.of<CounterProvider>(context).cour
              style: const TextStyle(fontSize: 50),
            ),
            ElevatedButton(
              onPressed: () {
                Navigator.pop(context);
              },
              child: const Text('Go to Home Screen'))
          ],)),
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.add),
        onPressed: () {
          context.read<CounterProvider>().add();
```

```
Text(
  //Nhưng cach khac
  context.watch<CounterProvider>().c
  // Provider.of<CounterProvider>(co
  //Provider.of<CounterProvider>(con
  style: const TextStyle(fontSize: 5
),,)),
floatingActionButton: FloatingActionBu
  child: const Icon(Icons.add),
  onPressed: () {
    context.read<CounterProvider>().
      //cach khac
      //Provider.of<CounterProvider>
    },,));}}
```

```
//cach khac
//Provider.of<CounterProvider>(context, li
},,));}}
```

Lưu ý khi muốn dùng **Provider.of**:

```
Provider.of<CounterProvider>(context,
listen: true = context.watch
listen: false = context.read
```

Provider dùng Consumer

Sử dụng **Consumer** để lắng nghe và tái sử dụng widget con khi trạng thái thay đổi

Notes

```
class MyConsumer extends StatelessWidget {
  const MyConsumer({super.key});
  @override
  Widget build(BuildContext context) {
    //Using Consumer
    return Consumer<MySettings>(builder: (context, mySettings, child) =>
    return Scaffold(
      appBar: AppBar(
        title: const Text('Provider Demo'),
        backgroundColor: mySettings.color,
      ),
      drawer: Drawer(child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () {
                mySettings.changeText();
                Navigator.pop(context);
                //auto out ra khoi drawer s
              },
              child: const Text('Change Text')),
            ElevatedButton(
              onPressed: () {
                mySettings.changeColor();
                Navigator.pop(context);
              },
              child: const Text('Change Color'))
          ],,)),
      body: Center(
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
```

```

        onPressed: () {
          mySettings.changeText();
        },
        child: const Text('Change Text')),
      Text('${mySettings.text}')
    ], ), ), ); } } } }

```

MultiProvider

Sử dụng tương tự Provider

Chỉ thêm hàm MultiProvider() để sử dụng thêm nhiều Provider

Notes

```

void main() {
  runApp(MultiProvider(
    providers: [
      ChangeNotifierProvider(create: (_) => MySettings()),
      ChangeNotifierProvider(create: (_) => Counter())
    ],
    child: const MyMultiProvider(),
  ));
}

```

GetX

.obs vào đối tượng muốn thay đổi giá trị và cập nhật lại giao diện

Wrap **Obx** với widget muốn nhận sự thay đổi do

Notes

```

class MyGetX extends StatelessWidget {
  MyGetX({super.key});
  //var count = 0.obs;
  RxInt count = 0.obs;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Counter App')),
      body: Center(
        child: Obx(() => Text('$count',
          style: TextStyle(fontSize: 50),
        )),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          count++;
          ///print ra console
          print(count);
        },
        child: Icon(Icons.add),
      ),
    );
  }
}

```

GetX(RouteManagement)

Tạo class kế thừa từ lớp

GetxController

.obs vào đối tượng muốn thay đổi giá trị và cập nhật lại giao diện

Notes

```

import 'package:get/get.dart';
class Counter extends GetxController {
  // var count = 0.obs;
  RxInt count = 0.obs;
  void add() {
    count++;
  }
}

```

Get.put(Counter()), Get.find()

nhận giá trị của đối tượng

```

class MainScreen extends StatelessWidget {
  MainScreen({super.key});
  final counter = Get.put(Counter());
}

```

Wrap **Obx** với widget muốn nhận sự thay đổi

Get.to(OtherScreen()) chuyển sang màn hình khác

```
//final Counter counter = Get.find();
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Counter App'),
    ),
    body: Center(
      child: Column(
        children: [
          Obx(() => Text('${counter.count}',
            ),),
          ElevatedButton(
            onPressed: () {
              Get.to(OtherScreen());
            },
            child: const Text('Go to other Screen')),
          ElevatedButton(
            onPressed: () {
              Get.to(const ThirdScreen());
            },
            child: const Text('Go to third Screen')),
        ],),),
    floatingActionButton: FloatingActionButton(
      onPressed: () {
        counter.add();
      },
      child: const Icon(Icons.add),
    ),);}}
```



SUMMARY:

Date: @November 5, 2019



SUMMARY: