

# GIAO DIỆN NGƯỜI DÙNG

Huỳnh Tuấn Anh  
Khoa CNTT-ĐHNT



## Nội dung

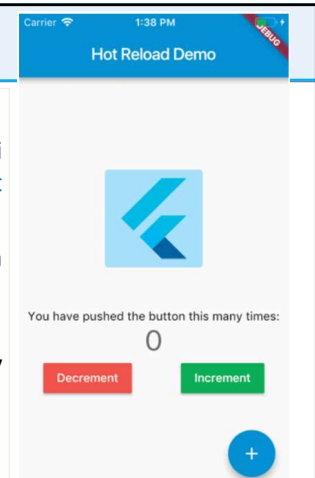
1. Giới thiệu về Widget
2. Một số Widget thông dụng
3. Layouts trong Flutter
4. Constraints
5. Assets và Image
6. Form Widget
7. Navigation và Routing

## 1. Giới thiệu về Widget



## Widget


- Mọi thứ trên giao diện đều là các **Widget**
- Mỗi **Widget** có thể được gắn với một trạng thái (state). Khi trạng thái của **widget** thay đổi, **widget** có thể được vẽ lại trên màn hình.
- Một widget có thể được bọc trong một widget cha để thừa hưởng các đặc trưng của widget cha.
  - VD: Bọc một **Text** bởi một **Container**
- Mỗi **Widget** thường có một thuộc tính **child** hay **children** và lập trình viên có thể gắn một hay nhiều **Widget** cho thuộc tính này.



## Widget – Hello World app!

```
void main() {
  runApp(
    Container(
      color: Colors.white,
      child: Center(
        child: Text(
          "Hello World!",
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 40,
            fontWeight: FontWeight.bold,
            color: Colors.black
          ),
        ),
      ),
    ),
  );
}
```

Run



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## Basic Widgets

- Text: Widget cho phép hiển thị chuỗi text có thể được định dạng trên màn hình ứng dụng.
  - VD:
 

```
Text(
    'Hello World!',
    textDirection: TextDirection.ltr,
    style: TextStyle(
      color: Colors.blue,
      fontSize: 40
    ),
  ),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

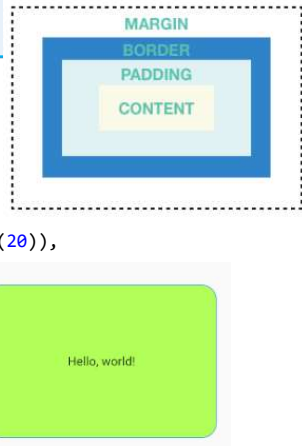
## Basic Widgets

- Row, Column: Thường được sử dụng để tạo các layout một cách linh hoạt.
  - Row: Được sử dụng để tạo layout sắp xếp các Widget theo chiều ngang.
  - Column: Được sử dụng để tạo layout sắp xếp các Widget theo chiều dọc.
- Stack: Được sử dụng để xếp các Widget chồng lên nhau. Widget đưa vào Stack sau cùng được hiển thị trên các Widget đưa vào trước.
  - Positioned: Widget Được sử dụng để bọc một widget con của Stack để chỉ định vị trí của widget con đó trong stack.
- Container: Cho phép tạo một thành phần hiển thị hình chữ nhật. Thường được sử dụng để bọc các Widget khác để xác định không gian hiển thị hay “Decorate” cho các widget này.
  - BoxDecoration: Thành phần dùng để Decorate cho Widget

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## Container – Ví dụ

```
Center(
  child: Container(
    width: 300,
    height: 200,
    decoration: BoxDecoration(
      color: Colors.LightGreenAccent,
      border: Border.all(color: Colors.blue),
      borderRadius: BorderRadius.all(Radius.circular(20)),
    ),
    child: Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  ),
),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## Statefull và Stateless Widget



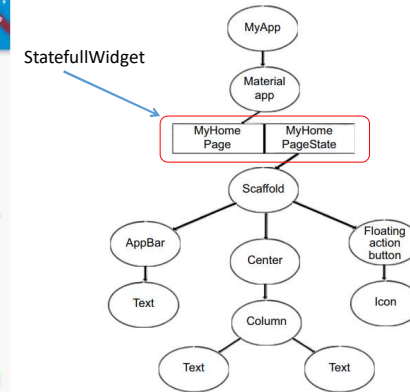
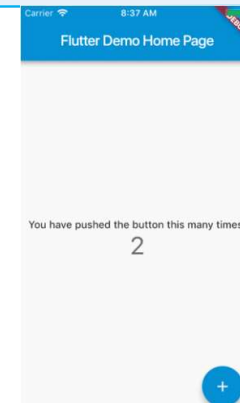
## Stateless Widget

- Widget không có trạng thái bên trong nào thay đổi trong thời gian tồn tại của nó.
- Widget này không quan tâm đến cấu hình và trạng thái mà nó hiển thị, nó có thể nhận dữ liệu hay cấu hình từ thành phần cha, nhưng tự nó không thể thay đổi dữ liệu và cấu hình của chính nó.
- Một Stateless Widget không thể tự phát sinh sự kiện để render lại chính mình.
- Không thay đổi khi người dùng tương tác với widget.
- Một số Stateless Widget: `Icon`, `IconButton`, `Text`
- Các Stateless Widget là lớp con của lớp `StatelessWidget`

## Stateful Widget

- Có trạng thái bên trong và có thể quản lý, tự thay đổi được trạng thái của chính mình.
- Có thể chứa nhiều `StatefulWidget` khác.
- Thay đổi khi người dùng tương tác với widget.
- Một số Stateful widget: `Checkbox`, `Radio`, `Slider`, `InkWell`, `Form`, `TextField`.
- Các Stateful widget là lớp con của lớp `StatefulWidget`
- Trạng thái của widget được lưu trữ trong đối tượng State tách biệt với `StatefulWidget`
  - Phương thức `setState()`: Bảo cho framework vẽ lại widget với trạng thái mới đã được cập nhật.

## Widget Tree



## StatefulWidget

- Mỗi thể hiện của một StatefulWidget liên quan đến hai class

Overrides the superclass method createState

```
class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
```

← Inherits from StatefulWidget

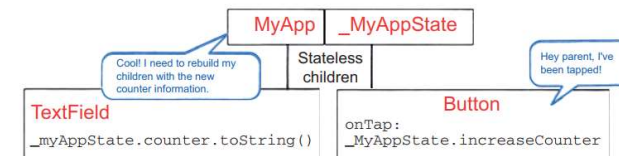
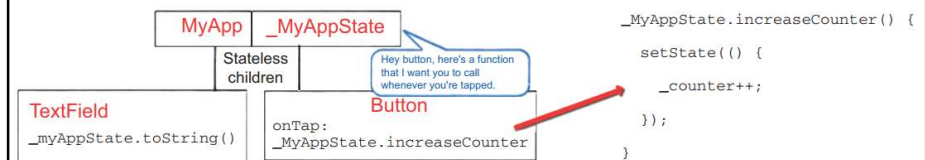
```
class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    // ..
  }
```

← Your state class inherits from the Flutter State object.

StatefulWidget's required build method

Every stateful widget must have a createState method that returns a State object.

## State class



## Phương thức initState

- Phương thức của lớp State dùng để khởi tạo trạng thái của widget khi đưa widget vào cây widget. Thông tin mà widget hiển thị lần đầu trên màn hình sẽ được khởi tạo trong phương thức này.

```
class FirstNameTextState extends State<FirstNameText> {
  String name;
```

```
  FirstNameTextState(this.name);
```

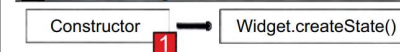
```
  @override
  initState() {
    super.initState();
    name = name.toUpperCase();
  }
```

The State.initState method is marked as mustCallSuper in the superclass. So, you must call the superclass implementation of initState in your overridden method.

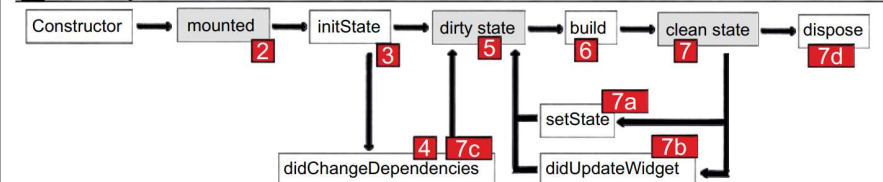
```
  Widget build(BuildContext context) {
    return Text(name);
  }
}
```

## StatefulWidget Lifecycle

### 1 StatefulWidget



### 2 State object



## BuildContext

- Đối tượng được Flutter cung cấp để:
  - Theo dõi toàn bộ widget tree, chứa thông tin vị trí của widget trong widget tree.
  - Là tham số của phương thức build, tham chiếu đến vị trí của widget trong cây.
  - Mỗi widget có một buildContext riêng của nó.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

17

## Widget Tree

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

18

## Widget Tree

- UI trong Flutter bao gồm nhiều Widget kết hợp với nhau hình thành nên một Widget Tree.
- Mỗi ứng dụng Flutter được biểu diễn bởi một Widget Tree trong đó mỗi node là một widget.
- Mỗi Widget có thuộc tính child chứa một widget con; hoặc thuộc tính children chứa một danh sách các widget con

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

19

## Widget Tree

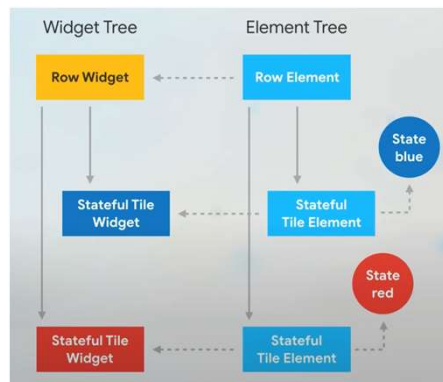
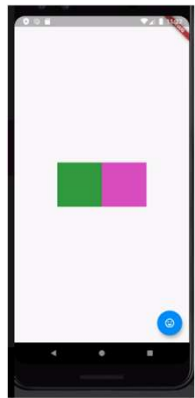
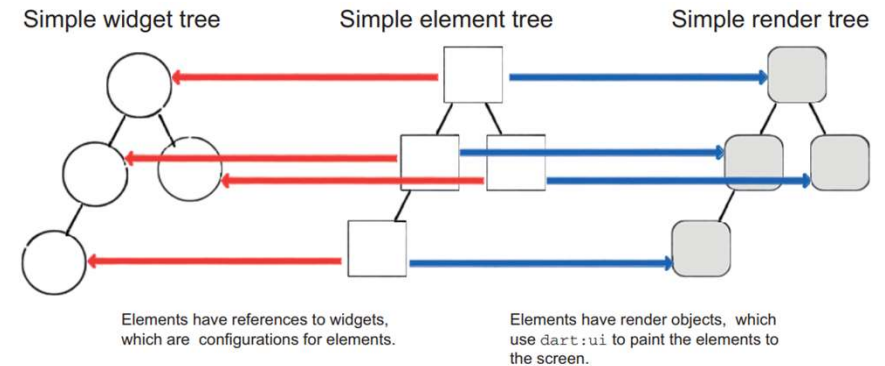
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

20

## Widget Tree , Element Tree

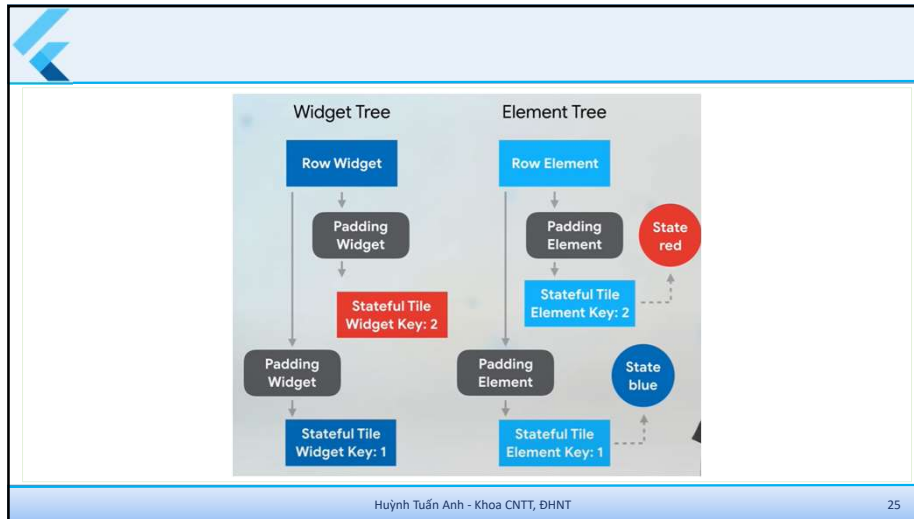
- Widget Tree: Gồm các widget dùng được dùng để xây dựng ứng dụng
  - Mỗi widget trong cây được render khi phương thức build của nó được gọi.
- Element: Các element được tạo ra bởi các widget. Mỗi khi một widget mới được build, Flutter framework sẽ gọi phương thức `Widget.createElement(this)` để tạo một element tương ứng. Mỗi element sẽ có một tham chiếu đến widget trong widget tree và widget này chính là cấu hình đầu tiên của element.
- Các element cũng có cây riêng của nó và được gọi là element tree.
- Element khác với widget vì nó không được rebuild mà chỉ được cập nhật.
  - Khi một Widget được rebuild hay một widget khác được chèn vào một vị trí nào đó, một element có thể thay đổi tham chiếu của nó đến widget

## Widget Tree , Element Tree



## Widget Key

- Trong trường hợp thay đổi/hoán đổi vị trí của các stateful widget, các widget này có thể không được tham chiếu đúng bởi các element với các state tương ứng. Trong trường hợp này ta có thể sử dụng các key để các element tham chiếu đúng đến các widget có cùng key với nó.
- Không cần sử dụng key cho stateless widget (SV giải thích)
- Các dạng của widget key: `ValueKey`, `ObjectKey`, `UniqueKey`, `PageStorageKey`, `GlobalKey`.
- Có thể tham chiếu đến các widget, state của các widget đó thông qua key của nó.



### Widget Key Type

- `ValueKey<T>` : Thường được sử dụng khi một widget dùng để hiển thị một nội dung được phân biệt với các nội dung khác bằng các id
- `ObjectKey`: Được sử dụng trong trường hợp Widget được sử dụng để hiển thị một nội dung được phân biệt với các nội dung khác bằng một tập các giá trị.
- `UniqueKey`: Sử dụng Unique key để đảm bảo rằng mỗi widget là duy nhất.
- `PageStorageKey`: được sử dụng để lưu và khôi phục các giá trị có thể tồn tại lâu hơn các widget

### Widget Key Type

- **GlobalKey:**
  - Cho phép một widget có thể thay đổi parent của nó ở bất kỳ vị trí nào trong ứng dụng mà vẫn không bị mất state của nó.
  - Ví dụ: Có thể sử dụng một CheckBox với một Global key trên nhiều trang. Global Key báo cho Flutter biết rằng các CheckBox trên các trang là một. Khi người dùng thay đổi trạng thái check của CheckBox ở một trang thì trạng thái của các CheckBox có cùng Key ở các trang khác cũng thay đổi theo.
  - Tuy nhiên, việc sử dụng Global Key để quản lý state là không được khuyến nghị theo Flutter Team bởi vì nó ảnh hưởng đến hiệu năng của hệ thống do nó làm cho các widget phụ thuộc trong cây widget rebuild.

## 2. Một số widget thông dụng

The slide displays the logos for Apple, Android, and Flutter, indicating the cross-platform nature of the framework.

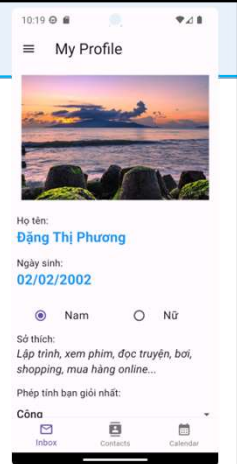


## Một số widget thông dụng

1. Container
2. Row, Column
3. Expanded
4. Stack
5. Image
6. Text, TextField
7. Icon
8. ElevatedButton, DropdownButton
9. Scaffold
10. ExpansionTitle,
11. ExpansionPanelList
12. AppBar
13. SafeArea
14. SingleChildScrollView
15. Builder
16. StatefulBuilder
17. FutureBuilder<T>: Widget được xây dựng để sử dụng với các nguồn dữ liệu không đồng bộ.
18. StreamBuilder<T>: Widget được xây dựng để làm việc với dữ liệu Stream

## MaterialApp

- Là một Widget trong Flutter cho phép xây dựng ứng dụng với thư viện Material Design của Google. Các thành phần của Material App bao gồm thành phần chính là một Scaffold. Trong Scaffold có các thành phần như:
  - AppBar
  - Body
  - BottomNavigationBar
  - FloatingActionButton
  - ...



## RaisedButton (thay bằng ElevatedButton)

- RaisedButton: Nút bấm, thường xử lý một sự kiện nào đó
- Constructor:
  - onPressed: một VoidCallback để xử lý sự kiện khi người dùng chạm vào nút.
  - child: Widget, thường là một Text để hiển thị tên của nút bấm

```
Widget build(BuildContext context) {
  return Container(
    color: Colors.white,
    child: Center(
      child: RaisedButton(
        child: const Text("Enable Button",
          style: TextStyle(fontSize: 20),),
        onPressed: () {},
      ),
    ),
  );
}
```



## Image

- Widget được sử dụng để hiển thị ảnh
- Hiển thị ảnh từ internet: Sử dụng Constructor: `Image.network(url)`
- Hiển thị ảnh từ local file: `Image.file(File)`
- Hiển thị ảnh từ thư mục asset: Sử dụng Constructor: `Image.asset(path_image)`
  - Cách thực hiện:
    - Tạo thư mục images trong thư mục gốc của project ứng dụng
    - Khai báo các image trong pubspec.yaml: `uses-material-design: true`
    - Ví dụ hiển thị pic1.jpg:
 


```
asset:
  - images/pic1.jpg
  - images/pic2.jpg
```



## Icon

- Widget hiển thị Icon
- Flutter cung cấp rất nhiều Icon đã được xây dựng sẵn.
- Ví dụ:

```
Row(
  mainAxisAlignment: MainAxisAlignment.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.black),
    Icon(Icons.star, color: Colors.black),
  ],
)
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

33

## Scaffold

- Cung cấp layout **Material Design** cơ bản, cho phép thêm các widget khác như **AppBar**, **BottomAppBar**, **FloatingActionButton**, **Drawer**, **SnackBar**, **BottomSheet**...
- Sử dụng với **MaterialApp**



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

34

## Scaffold – Một số thuộc tính cơ bản

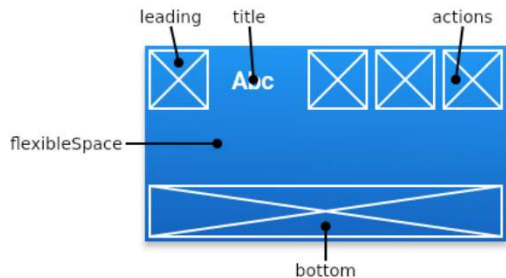
- body**: Widget thể hiện giao diện chính của Scaffold.
- appBar**: PreferredSizeWidget Thường là một AppBar hay TabBar của một màn hình.
- drawer**: Widget Biểu diễn một Drawer ở phía bên trái.
- endDrawer**: Widget Biểu diễn một Drawer ở phía bên phải.
- bottomNavigationBar**: Widget biểu diễn thanh điều hướng ở dưới màn hình.
  - Plugin: `bottom_navy_bar`, `convex_bottom_bar`
- floatingActionButton**: Widget biểu diễn một nút Floating button thường ở góc dưới bên phải của màn hình.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

35

## AppBar

- Được sử dụng cho thuộc tính `appBar` của **Scaffold**



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

36


## AppBar – Constructor

```
AppBar(
  {Key key,
  Widget leading,
  bool automaticallyImplyLeading: true,
  Widget title,
  List<Widget> 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,
  double leadingWidth}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## AppBar

```
appBar: AppBar(
  title: Text('Hello Flutter!'),
  leading: Icon(Icons.menu),
  actions: <Widget>[
    IconButton(icon: Icon(Icons.share), onPressed: ()=>{}),
    IconButton(icon: Icon(Icons.call), onPressed: ()=>{}),
  ],
),
```

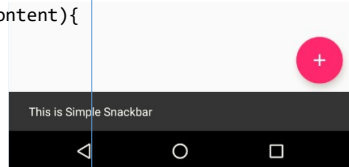


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## SnackBar

- Widget, là một thành phần của scaffold hiển thị thông báo nhanh cho người sử dụng. Theo mặc định, SnackBar nằm ở phía dưới cùng của màn hình.
- Các thuộc tính chính của SnackBar
  - content: Widget hiển thị nội dung của SnackBar
  - duration: Chỉ định thời gian hiển thị của SnackBar
  - action: SnackBarAction, hành động khi người dùng nhấn vào widget này trên SnackBar

```
void showSnackBar(BuildContext context, String content){
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(content),
      duration: Duration(seconds: 10),
    ));
}
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## SingleChildScrollView


- Thêm khả năng cuộn ngang/dọc cho một widget

```
const SingleChildScrollView(
  {Key key,
  Axis scrollDirection: Axis.vertical,
  bool reverse: false,
  EdgeInsetsGeometry padding,
  bool primary,
  ScrollPhysics physics,
  ScrollController controller,
  Widget child,
  DragStartBehavior dragStartBehavior: DragStartBehavior.start,
  Clip clipBehavior: Clip.hardEdge,
  String restorationId}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## SingleChildScrollView – ví dụ

```
body: SafeArea(
  child: SingleChildScrollView(
    child: Column(
      children: [
        Container(height: 300,
          color: Colors.yellow,),
        Container(height: 300,
          color: Colors.blue[300],),
        Container(height: 300,
          color: Colors.green[300],)
      ],
    ),
  ),
),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 41

## Expanded

- Widget giúp widget con của nó chia phần diện tích còn lại của Layout (Row hay Column)

```
body: Center(
  child: Row(
    children: [
      _getLoiChao("Hello", Colors.blue[100]),
      _getLoiChao("Chào", Colors.red[100], 2),
      _getLoiChao("Привет", Colors.green[100])
    ],
  ),
),
```

```
Widget _getLoiChao(String st, Color color, [int flex=1]) {
  return Expanded(flex: flex,
    child: Container(height: 100, color: color,
      child: Center(
        child: Text(st, style: TextStyle(fontSize: 20)),
      ),
    ));
}
```




Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 42

## TextField

- Widget cho phép user nhập chuỗi text từ bàn phím
- Sử dụng TextEditingController để lấy/thiết lập nội dung cho TextField

```
body: Center(
  child: Padding(
    padding: const EdgeInsets.all(5.0),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        TextField(
          controller: nameEditingController,
          decoration: InputDecoration(
            hintText: "Nhập tên vào đây",
            labelText: "Tên:"
          ),
        ),
        RaisedButton(onPressed: () => _showAlertDialog(context),
          child: Text("Submit"),
        ),
      ],
    ),
  ),
),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 43

```
void _showAlertDialog(BuildContext context){
  AlertDialog dialog = AlertDialog(
    title: Text("Xác nhận"),
    content: Text(nameEditingController.text),
    actions: [
      FlatButton(onPressed: () => Navigator.pop(context),
        child: Text("OK"))
    ],
  );
  showDialog(context: context,
    builder: (context) => dialog,
  );
}
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 44

### 3. Layout trong Flutter



### Layout trong Flutter

- Mọi thứ trong Flutter đều là Widget và các **Layout** trong Flutter cũng là các Widget.
- Widget rõ: Được nhìn thấy như: **Image, Icon, Text, Button..**
- Widget Layout: Không được nhìn thấy như: **Row, Column, Stack, Grid, Constraint...**
- Các layout widget được sử dụng để trình bày các widget rõ trên giao diện
  - Có thể lồng các layout widget lồng nhau để trình bày các widget rõ. Ví dụ con thể lồng Row vào Column hoặc ngược lại để trình bày các widget rõ theo chiều dọc/ngang

### Layout một widget

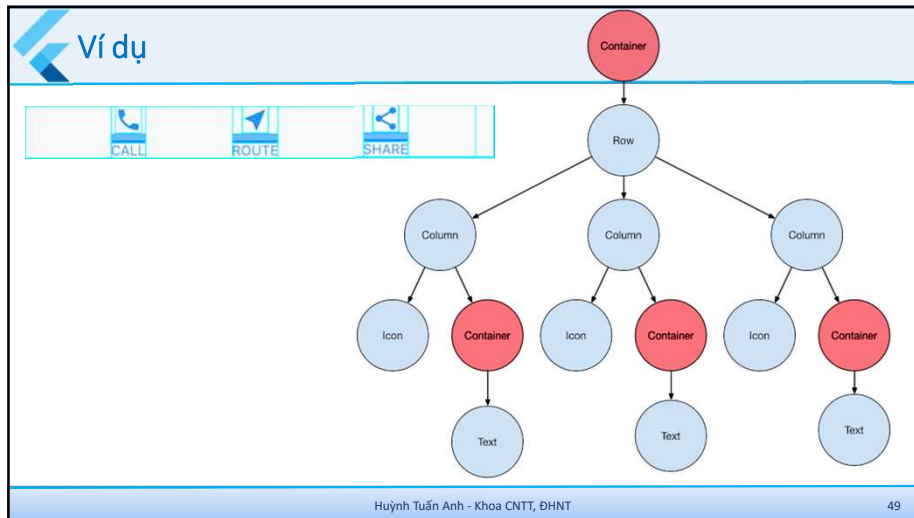
- Trong Flutter, các thuộc tính về layout của widget cha sẽ truyền lại cho widget con. Do đó, ta có thể sử dụng các Layout Widget để gói các Widget rõ để truyền các đặc tính về layout của widget cha cho widget con.
- Đặt các widget rõ trong một **layout widget** phù hợp với các ràng buộc hiển thị. Một số các ràng buộc hiển thị:
  - Cách thức các widget được sắp xếp: Theo hàng, cột, lưới, danh sách. Các Widget thường được sử dụng là: **Row, Column, GridView, ListView**
  - Các ràng buộc về hiển thị: Căn lề, chứa lề... Các Widget thường được sử dụng là: **Container, Padding, Center,**
  - Các ràng buộc về phân chia không gian hiển thị: sử dụng **Expanded**, thường được sử dụng cùng với **Column, Row** để phân chia không gian hiển thị cho các widget theo các chiều dọc/ngang bằng thuộc tính **flex**

### Ví dụ



Có thể hình dung hình trên bao gồm một dòng có 3 cột. Mỗi cột gồm một Icon và một chuỗi Text





49

### Tạo một widget rõ

- Phương pháp: Gọi hàm khởi tạo của widget cần tạo:
- Tạo một Text widget:
 

```
Text('Hello World'),
```
- Tạo một Image widget:
 

```
Image.asset(
  'images/lake.jpg',
  fit: BoxFit.cover,
),
```
- Tạo một Icon widget:
 

```
Icon(
  Icons.star,
  color: Colors.red[500],
),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

50

### Đưa widget rõ vào layout widget

- Tất cả các layout widget đều có thuộc tính:
  - child**: Có thể nhận một widget con. Ví dụ: Container, Center, Expanded... Hoặc
  - children**: Có thể nhận một danh sách các widget con. Ví dụ: Row, Column, Stack...
- Đưa widget rõ vào layout widget: Truyền *widget rõ* cho thuộc tính **child** hoặc **children** của layout widget.
- Ví dụ: Để dòng Text hiển thị ở chính giữa một không gian cho trước nào đó, ta sử dụng widget Center để gói widget Text.

```
Center(
  child: Text('Hello World'),
),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

51

### Thêm layout widget vào một trang (page)

- Trong ứng dụng flutter, mỗi màn hình được xem là một trang (page).
- Bản thân ứng dụng Flutter cũng là một widget, hầu hết các widget đều có phương thức build. Việc tạo và trả về một widget trong ứng dụng sẽ hiển thị widget đó.
- Material Apps**: Mỗi màn hình là một Scaffold widget
  - Nên sử dụng các widget trong **Material Library** với **Material App**.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

52

## Material Apps

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter layout demo',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter layout demo'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 53

## None Material App

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(color: Colors.white),
      child: Center(
        child: Text(
          'Hello World',
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 32,
            color: Colors.black87,
          ),
        ),
      ),
    );
  }
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 54

## Sắp xếp các widget theo chiều dọc và theo chiều ngang

- Sử dụng **Row** và **Column** để thực hiện việc sắp xếp các widget theo chiều ngang hoặc chiều dọc

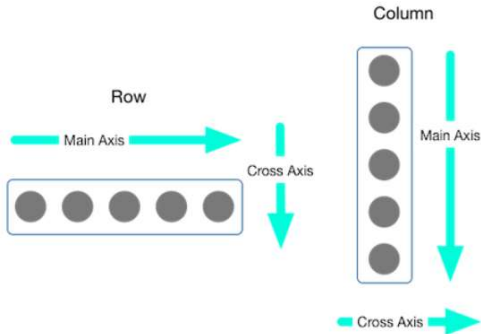
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 55

## Sắp xếp các widget theo chiều dọc và theo chiều ngang

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 56

## Sắp các widget

- **mainAxisAlignment**: Cách các widget con được sắp theo trục chính
- **crossAxisAlignment**: Cách các widget được sắp theo trục chéo
- Các sắp các widget: **start**, **center**, **end**, **spaceEvenly**, **spaceBetween**, **spaceAround**.




Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

57

## Sắp các widget

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```



App source: [row\\_column](#)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

58

## Expanded: Phân bổ không gian hiển thị cho các widget

- Khi một layout lớn hơn kích thước của thiết bị → lỗi hiển thị.
- **Expanded**: Widget cho phép widget con của nó được điều chỉnh kích thước để hiển thị vừa với kích thước của hàng hoặc cột.

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```



App source: [sizing](#)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

59

## Expanded: Phân bổ không gian hiển thị cho các widget

- Sử dụng thuộc tính **flex** để xác định tỉ lệ phân chia không gian của layout

```
Row(
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Expanded(
      child: Image.asset('images/pic1.jpg'),
    ),
    Expanded(
      flex: 2,
      child: Image.asset('images/pic2.jpg'),
    ),
    Expanded(
      child: Image.asset('images/pic3.jpg'),
    ),
  ],
);
```



App source: [sizing](#)

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

60



## Flexible

- Flexible cho phép cho phần tử con của Row, Column hoặc Flex khả năng mở rộng linh hoạt để lấp đầy khoảng trống có sẵn theo trục chính (ví dụ: theo chiều ngang cho Row hoặc theo chiều dọc cho Column), nhưng, không giống như Expanded, Flexible không yêu cầu các phần tử con phải lấp đầy khoảng trống có sẵn.

```
Flexible({
  Key? key,
  int flex = 1,
  FlexFit fit = FlexFit.loose,
  required Widget child})
```

**FlexFit.tight** = Wants to fit tight into parent taking as much space as possible.  
**FlexFit.loose** = Wants to fit loose into parent taking as little space as possible for itself.

## Nén khoảng cách các Widget theo trục chính của layout

- Mặc định Row hoặc Column chiếm hết không gian có thể được theo trục main. Sử dụng thuộc tính `mainAxisSize` để nén các khoảng cách các widget con

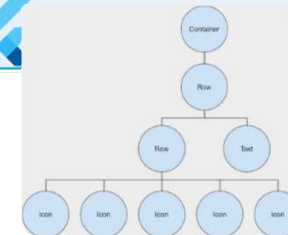
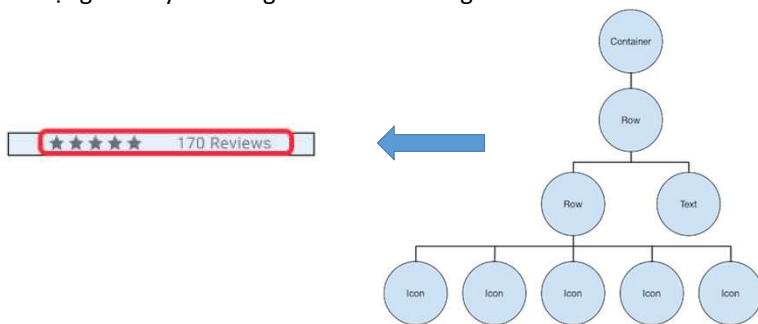
```
Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.black),
    Icon(Icons.star, color: Colors.black),
  ],
)
```



App source: [pavlova](#)

## Row và Column lồng nhau

- Mọi thứ trong Flutter đều là Widget kể cả các Layout Widget. Do đó ta có thể sử dụng các Layout Widget như là các widget con.



```
var stars = Row(
  mainAxisSize: MainAxisSize.min,
  children: [
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.green[500]),
    Icon(Icons.star, color: Colors.black),
    Icon(Icons.star, color: Colors.black),
  ],
);
```

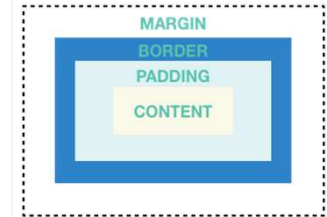
```
final ratings = Container(
  padding: EdgeInsets.all(20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
    children: [
      stars,
      Text(
        '170 Reviews',
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.w800,
          fontFamily: 'Roboto',
          letterSpacing: 0.5,
          fontSize: 20,
        ),
      ),
    ],
  ),
);
```

## Các layout widget phổ biến

- Standar widgets
  - Container
  - ListView
  - GridView
  - Stack
- Material widgets
  - Card
  - ListTile

## Container

- Sử dụng Container để tách cách các widget bằng các thuộc tính **padding**, **border** hay **margin**.
- Xác định kích thước vùng không gian hiển thị bằng hai thuộc tính **width**, **height** cho các widget con bên trong.
- Sử dụng Container khi cần:
  - Thêm padding, lề, đường viền.
  - Thay đổi màu nền hay hình ảnh
  - Chứa một widget con đơn lẻ, và widget này có thể là Row, Column, hay thậm chí là gốc của cây widget.
  - Tạo vùng không gian có kích thước cụ thể cho các widget con bên trong.



## Container

```
Widget _buildImageColumn() => Container(
  decoration: BoxDecoration(
    color: Colors.black26,
  ),
  child: Column(
    children: [
      _buildImageRow(1),
      _buildImageRow(3),
    ],
  ),
);
```



## Container

```
Widget _buildDecoratedImage(int imageIndex) => Expanded(
  child: Container(
    decoration: BoxDecoration(
      border: Border.all(width: 10, color: Colors.black38),
      borderRadius: const BorderRadius.all(const Radius.circular(8)),
    ),
    margin: const EdgeInsets.all(4),
    child: Image.asset('images/pic$imageIndex.jpg'),
  ),
);

Widget _buildImageRow(int imageIndex) => Row(
  children: [
    _buildDecoratedImage(imageIndex),
    _buildDecoratedImage(imageIndex + 1),
  ],
);
```

## GridView

- Sử dụng **GridView** để sắp xếp các widget dưới dạng *danh sách hai chiều*. **GridView** cung cấp hai danh sách được tạo sẵn hoặc bạn có thể tạo lưới tùy chỉnh của riêng mình. Khi **GridView** phát hiện thấy nội dung của nó quá dài để vừa với hộp kết xuất, nó sẽ tự động cuộn.
- Xây dựng một **GridView** tùy chỉnh hoặc sử dụng một trong các grid được cung cấp sẵn:
  - GridView.count**: Chỉ định *số lượng cột* trong grid.
  - GridView.extent**: Cho phép chỉ định *độ rộng tối đa* (theo pixel) của một ô lưới.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## Ví dụ: GridView.extent

```
body: GridView.extent(
  maxCrossAxisExtent:150,
  padding: EdgeInsets.all(5),
  mainAxisSpacing: 5,
  crossAxisSpacing: 5,
  children: List.generate(100,
    (index) => Container(
      decoration: BoxDecoration(
        border: Border.all(color: Colors.blue)
      ),
      child: Center(
        child: Text("${index+1}",
          style: TextStyle(fontSize: 20),),
      ),
    ),
  ),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## Ví dụ: GridView.count

```
body: GridView.count(
  crossAxisCount: 3,
  children: List.generate(100,
    (index) => Container(
      decoration: BoxDecoration(
        border: Border.all(color: Colors.blue)
      ),
      child: Center(
        child: Text('${index+1}',
          style: TextStyle(fontSize: 20),),
      ),
    ),
  ),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT


## ListView

- Tương tự như Column, nhưng tự động cuộn khi các widget con vượt quá kích thước của render box.
- ListView**:
  - Là một Column đặt biệt để tổ chức một danh sách các box.
  - Các widget con có thể được sắp theo chiều dọc hoặc chiều ngang.
  - Tự động cuộn khi nội dung trong danh sách vượt quá kích thước hiển thị.
  - Không cần phải cấu hình nhiều như Column, nhưng rất hỗ trợ cuộn rất dễ dàng
- ListView constructor**
  - ListView.builder**
  - ListView.separated**

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

## ListTile

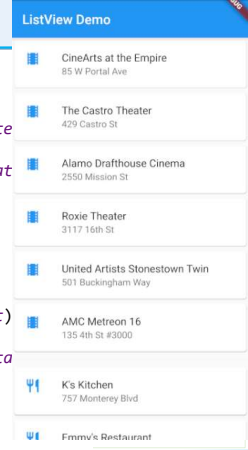
- Là một row widget đặc biệt trong gói thư viện Material. ListTile gồm 3 phần:
  - Trailing icon
  - Title
  - Subtitle
- ListTile thường được sử dụng làm các item của ListView



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 73

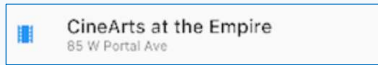
## ListView

```
Widget build(BuildContext context) {
  return ListView(
    children: <Widget>[
      _tile('CineArts at the Empire', '85 W Portal Ave', Icons.theat
      _tile('The Castro Theater', '429 Castro St', Icons.theaters),
      _tile('Alamo Drafthouse Cinema', '2550 Mission St', Icons.theat
      _tile('Roxie Theater', '3117 16th St', Icons.theaters),
      _tile('United Artists Stonestown Twin', '501 Buckingham Way',
        Icons.theaters),
      _tile('AMC Metreon 16', '135 4th St #3000', Icons.theaters),
      Divider(),
      _tile('K's Kitchen', '757 Monterey Blvd', Icons.restaurant),
      _tile('Emmy's Restaurant', '1923 Ocean Ave', Icons.restaurant)
    ],
    _tile('Chaiya Thai Restaurant', '272 Claremont Blvd', Icons.rest
    _tile('La Ciccia', '291 30th St', Icons.restaurant),
  ],
);
}
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 74

```
ListTile _tile(String title, String subtitle, IconData icon) => ListTile(
  title: Text(title,
    style: TextStyle(
      fontWeight: FontWeight.w500,
      fontSize: 20,
    )),
  subtitle: Text(subtitle),
  leading: Icon(
    icon,
    color: Colors.blue[500],
  ),
);
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 75

## Stack

- Được sử dụng để xếp chồng các widget.



```
Widget _buildStack() => Stack(
  alignment: const Alignment(0.6, 0.6),
  children: [
    CircleAvatar(
      backgroundImage: AssetImage('images/pic.jpg'),
      radius: 100,
    ),
    Container(
      decoration: BoxDecoration(
        color: Colors.black45,
      ),
      child: Text(
        'Mia B',
        style: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
    ),
  ],
);
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 76

## Card


- Một widget trong thư viện Material.
- Được sử dụng để hiển thị các thông tin ngắn gọn có liên quan với nhau được chứa bởi mọi các widget. Thông thường, Card được sử dụng với ListTile. Không thể cuộn nội dung trên Card.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 77




## Ví dụ

```
Widget _buildCard() =>
  SizedBox(
    height: 210,
    child: Card(
      child: Column(
        children: [ ListTile(
          title: Text('1625 Main Street',
            style: TextStyle(fontWeight: FontWeight.w500)),
          subtitle: Text('My City, CA 99984'),
          leading: Icon( Icons.restaurant_menu, color: Colors.blue[500]
        ),
      ),
    ),
  ),
  ...
```

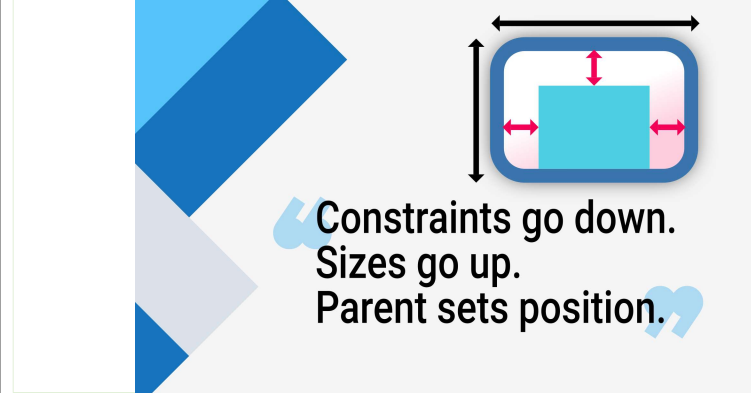


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 78

## 5. Constraints

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 79



Constraints go down.  
Sizes go up.  
Parent sets position.

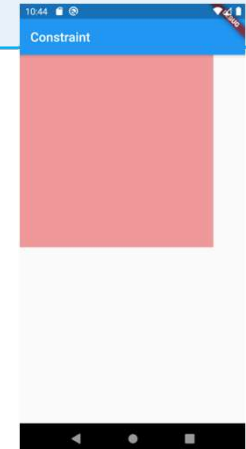
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 80

## Constraints

- Một widget nhận các ràng buộc riêng từ widget cha của nó. Ràng buộc chỉ là một tập hợp 4 giá trị double: **minimum** and **maximum width**, và **minimum** và **maximum height**.
- Widget cha duyệt qua danh sách con của nó và “nói” cho từng widget con constraint của chúng (có thể khác nhau đối với mỗi widget). Và sau đó “hỏi” kích thước mong muốn của từng widget con.
- Sau đó widget cha định vị các widget con của nó (theo trục x và theo trục y).
- Cuối cùng widget con “nói” cho widget cha về kích thước của nó (tất nhiên là thỏa mãn ràng buộc ban đầu)

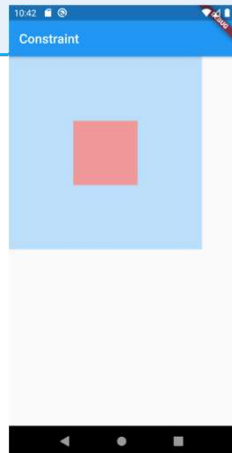
## Ví dụ 1

```
class MyConstraint extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Constraint"),
      ),
      body: Container(
        width: 300, height: 300,
        color: Colors.blue[100],
        child: Container(
          width: 100, height: 100,
          color: Colors.red[100],
        ),
      ),
    );
  }
}
```



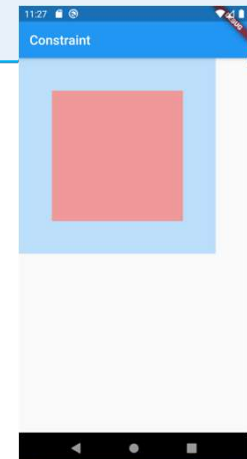
## Ví dụ 2

```
class MyConstraint extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Constraint"),
      ),
      body: Container(
        width: 300, height: 300,
        color: Colors.blue[100],
        child: Center(
          child: Container(
            width: 100, height: 100,
            color: Colors.red[200],
          ),
        ),
      ),
    );
  }
}
```



## Ví dụ 3

```
class MyConstraint2 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Constraint"),
      ),
      body: Container(
        width: 300, height: 300,
        color: Colors.blue[100],
        child: Center(
          child: Container(
            constraints: BoxConstraints(
              maxWidth: 200, maxHeight: 200,
              minWidth: 100, minHeight: 100
            ),
            color: Colors.red[200],
          ),
        ),
      ),
    );
  }
}
```



## Ràng buộc chặt và ràng buộc lỏng lẻo

- Ràng buộc chặt chẽ: Cung cấp một ràng buộc chính xác và duy nhất.

```
BoxConstraints.tight(Size size)
: minWidth = size.width,
  maxWidth = size.width,
  minHeight = size.height,
  maxHeight = size.height;
```

- Ràng buộc lỏng lẻo: Cung cấp một ràng buộc với các tham số ràng buộc trong một khoảng nào đó.

```
BoxConstraints.loose(Size size)
: minWidth = 0.0,
  maxWidth = size.width,
  minHeight = 0.0,
  maxHeight = size.height;
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 85

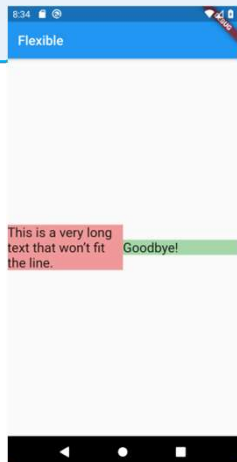
## Expanded, Flexible widget

- Flexible** widget: Cho phép widget con của nó có chiều rộng bằng hoặc nhỏ hơn chính bản thân của **Flexible**.
  - thuộc tính `fit = FlexFit.tight`: Flexible thực hiện chức năng của một Expanded
  - thuộc tính `fit = FlexFit.loose`: là giá trị default của thuộc tính `fit`
- Expanded**: Bắt buộc widget con của nó có chiều rộng bằng với chính bản thân của **Expanded**
- Cả hai widget này đều *bỏ qua chiều rộng của widget con* khi tự xác định kích thước của mình.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 86

## Expanded


```
body: Center(
  child: Row(
    children: [
      Expanded(child: Container(color: Colors.red[200],
        child: Text('This is a very long text that '
          'won't fit the line.',
          style: TextStyle(fontSize: 20)),
      ),
      Expanded(child: Container(color: Colors.green[200],
        child: Text('Goodbye!',
          style: TextStyle(fontSize: 20)),
      ),
    ],
  ),
),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 87

## Flexible

```
body: Center(
  child: Row(
    children: [
      Flexible(child: Container(color: Colors.red[200],
        child: Text('This is a very long text that '
          'won't fit the line.',
          style: TextStyle(fontSize: 20)),
      ),
      Flexible(child: Container(color: Colors.green[200],
        child: Text('Goodbye!',
          style: TextStyle(fontSize: 20)),
      ),
    ],
  ),
),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 88



## 6. Asset và Image



### Asset

- **Asset:** Còn gọi là tài nguyên, là các tệp được đóng gói và triển khai cùng với ứng dụng và có thể được truy cập trong thời gian chạy.
- Các dạng Asset: Dữ liệu tĩnh
  - JSON File
  - Image, Icon
  - Các file cấu hình
- Mỗi Asset được định danh bởi một đường dẫn tương minh được khai báo ở tập tin [pubspec.yaml](#). Trật tự khai báo các asset không quan trọng.
- Các asset sẽ được build cùng với ứng dụng.

### Khai báo Asset

- Sử dụng file `pubspec.yaml`, khai báo như sau:

```
flutter:
  assets:
    - assets/my_icon.png
    - assets/background.png
```

Thư mục assets nằm trong thư mục gốc của project

- Để bao gồm tất cả các Asset trong một thư mục, khai báo tên thư mục với ký tự `"/"` đứng ở cuối

```
flutter:
  assets:
    - directory/
    - directory/subdirectory/
```

### Nạp (load) asset vào ứng dụng

- Có 2 phương thức chính trên asset bundle cho phép nạp string/text hoặc image/binary asset trên một logical key
  - `loadString(String path)`: Nạp string/text
  - `load(String path)`: Nạp image/binary

## Nạp text asset

- Mỗi ứng dụng Flutter có một đối tượng `rootBundle` để truy cập asset bundle. Có thể nạp asset một cách trực tiếp bằng cách sử dụng đối tượng tĩnh toàn cục `rootBundle` từ: `package:flutter/services.dart`.
  - `rootBundle` thường được sử dụng bên ngoài một widget context để nạp trực tiếp asset

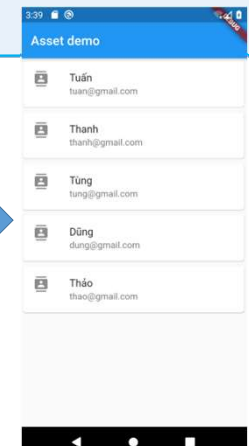
```
import 'dart:async' show Future;
import 'package:flutter/services.dart' show rootBundle;

Future<String> loadAsset() async {
  return await rootBundle.loadString('assets/config.json');
}
```

## Ví dụ

- Nạp file Json từ asset

```
[
  {
    "name": "Tuấn",
    "email": "tuan@gmail.com"
  },
  {
    "name": "Thanh",
    "email": "thanh@gmail.com"
  },
  {
    "name": "Tùng",
    "email": "tung@gmail.com"
  },
  {
    "name": "Dũng",
    "email": "dung@gmail.com"
  },
  {
    "name": "Thảo",
    "email": "thao@gmail.com"
  }
]
```



## Code

```
import 'dart:convert';
import 'package:flutter/services.dart' show rootBundle;
```

```
class User {
  final String name;
  final String email;
  User({this.name, this.email});
```

```
  User.fromJson(Map<String, dynamic> json):
    name = json['name'],
    email = json['email'];
```

```
  Map<String, dynamic> toJson()
  {
    return {
      'name': name,
      'email': email
    };
  }
}
```

Hai phương thức cần có để serialization/deserialization

```
Future<List<User>> fetchUserFromJson(String path) async{
  String jsonStr = await rootBundle.loadString(path);
  List<User> users;
  var list = json.decode(jsonStr) as List;
  users = list.map((item)=> User.fromJson(item)).toList();
  return users;
}
```

→ Đọc text file

↓  
Chuyển chuỗi Json thành danh sách các đối tượng User

## Page hiển thị: Stateful widget

```
class _UserFromAssetPageState extends State<UserFromAssetPage> {
  Future<List<User>> users;
  @override
  void initState() {
    super.initState();
    users = fetchUserFromJson('asset/jsons/users.json');
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Asset demo')),
      body: Container(
        child: FutureBuilder<List<User>>(
          future: users,
          builder: (context, snapshot) {
            //...
          },
        ),
      ),
    );
  }
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

97

```
body: Container(
  child: FutureBuilder<List<User>>(
    future: users,
    builder: (context, snapshot) {
      if(snapshot.hasData) {
        return ListView(
          children: List.generate(snapshot.data.length,
            (index) => Card(
              child: ListTile(
                title: Text(snapshot.data[index].name),
                subtitle: Text(snapshot.data[index].email),
                leading: Icon(Icons.contacts),
              ),
            ),
          ),
        );
      }
      else if(snapshot.hasError)
        return Text('Lỗi đọc dữ liệu');
      else
        return CircularProgressIndicator();
    },
  ),
),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

98

## Nạp Images

- Flutter có thể nạp các image với độ phân giải phù hợp với thiết bị sử dụng
- Khai báo các asset với các ratio khác nhau trong tệp pubspec.yaml theo cấu trúc thư mục:

```
uses-material-design: true
asset:
  - icons/heart.png
  - icons/1.5x/heart.png
  - icons/2.0x/heart.png
```

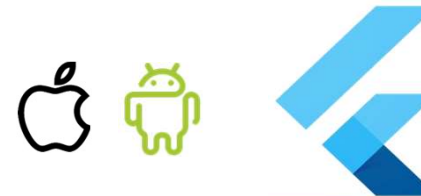
- Nạp image:

```
Widget build(BuildContext context) {
  return Image(image: AssetImage('icons/heart.png'));
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

99

## 3. Form Widget



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

100

## Nội dung

- Form Widget
- GlobalKey<FormState>
- FormField Widget
- TextFormField
- DropDownButtonFormField

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 101

## Form

- Là một container để nhóm các nhiều **FormField** cùng với nhau.
- Mỗi trường trong Form nên được bọc bởi **FormField** Widget với **Form** Widget là *ancestor* chung của tất cả các trường.
- Mỗi Form có một đối tượng **FormState** được khai báo thông qua **GlobalKey**
- Có thể gọi các phương thức trên **FormState** để lưu, đặt lại (reset) hoặc xác thực lại từng **FormField** con của **Form**.
- Để lấy **FormState**
  - Gọi **Form.of** với context có ancestor là Form; hoặc
  - Chuyển một **GlobalKey** cho thuộc tính **key** trong phương thức khởi tạo của **Form** và gọi **GlobalKey.currentState**.

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 102

## Form

- key: GlobalKey<FormState>
- autovalidateMode: **AutovalidateMode**
  - AutovalidateMode.disabled** (default)
  - AutovalidateMode.onUserInteraction**: Form field chỉ tự động validate sau khi nội dung thay đổi
  - AutovalidateMode.always**: Có thể tự động validate mà không cần sự tương tác của user
- onWillPop: **WillPopCallback**
  - Cho phép Form phủ quyết nỗ lực của người dùng để loại bỏ ModalRoute có chứa Form.
  - Return: Future<bool>**: "form route" sẽ không được pop

```
const Form({
  1. Key key,
  2. @required Widget child,
  3. WillPopCallback onWillPop,
  4. VoidCallback onChanged,
  5. AutovalidateMode autovalidateMode})
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 103

## GlobalKey<FormState>

- Đối tượng dùng để kiểm soát **Form** thông qua **FormState**.
- Các phương thức trên **FormState**
  - validate()**: nếu **AutovalidateMode.disabled** sẽ không xảy ra một validation nào cho đến khi phương thức **validate()** được gọi
  - save()**: Gọi các *callback* **onSaved** của các **FormField**
  - reset()**

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 104

## FormField Widget

- Form cung cấp `FormState` để làm việc với các trường đầu vào có liên quan với nhau. Để Form quản lý được các trường đầu vào này, chúng phải là các `FormField` Widget.
- `FormField` Widget được dùng để bọc các widget khác, giúp chúng có thể được quản lý bởi Form.
  - Ví dụ: Có thể sử dụng `CheckBox` trong Form nhưng nó phải được bọc trong `FormField`

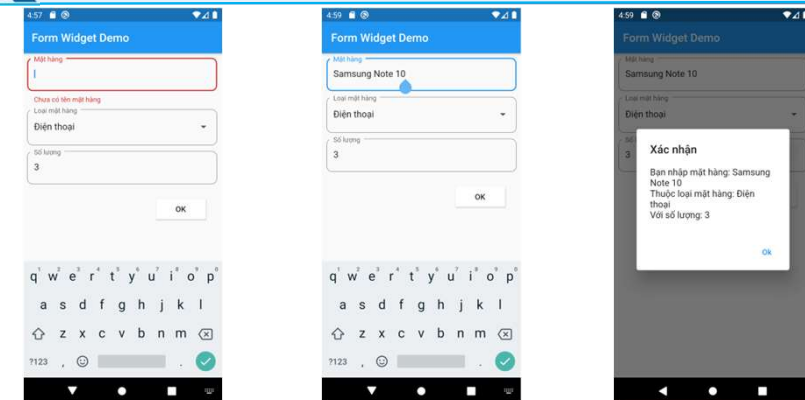
## TextFormField

- Là một `FormField` đặt biệt đã bọc sẵn một `TextField`
- `FormField` không cần thiết phải sử dụng với một Form làm ancestor. Trong trường hợp này cần phải chỉ định một `GlobalKey` cho `TextFormField` và sử dụng `GlobalKey.currentState` để lưu hoặc reset `FormField`

## DropDownButtonFormField

- Là một widget tiện ích đã được bọc sẵn `DropDownButton` trong một `FormField`
- Các thuộc tính:
  - @required `List<DropDownMenuItem<T>> items`,
  - @required `this.onChange`,
  - `FormFieldSetter<T> onSaved`,
  - `FormFieldValidator<T> validator`,

## Ví dụ




## App

```
class MyForm extends StatelessWidget {
  GlobalKey<FormState> _formState = GlobalKey<FormState>();
  MathHang mh = MathHang();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(...), // AppBar
      body: Form(...), // Form
    ); // Scaffold
  }
}

List<String> loaiMHs = ['Tivi', 'Điện thoại', 'Laptop'];
class MathHang{
  String tenMH, loaiMH;
  int soLuong;
  MathHang({this.tenMH, this.loaiMH, this.soLuong});
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 109

```
body: Form(
  key: _formState,
  autovalidateMode: AutovalidateMode.disabled,
  child: Padding(
    padding: const EdgeInsets.all(8.0),
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        TextFormField(...), // TextFormField
        SizedBox(height: 10,),
        DropdownButtonFormField<String>(...), // Input
        SizedBox(height: 10,),
        TextFormField(...), // TextFormField
        SizedBox(height: 20,),
        Row(
          children: [
            Expanded(child: SizedBox(), flex: 1,),
            RaisedButton(...), // RaisedButton
            SizedBox(width: 20,)
          ],
        ), // Row
      ],
    ),
  ),
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 110

## TextFormField

```
TextFormField(
  decoration: InputDecoration(
    labelText: 'Mặt hàng',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10))
  ),
  onSave: (newValue) {mh.tenMH=newValue;},
  validator: (value) => value.isEmpty ? "Chưa có tên mặt hàng" : null,
),

String _validateSoluong(String value) {
  int sl = value.isEmpty ? 0 : int.tryParse(value);
  return sl<=0 ? "Số lượng mặt hàng phải lớn hơn 0" : null;
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 111

## DropdownButtonFormField

```
DropdownButtonFormField<String>(
  items: loaiMHs.map((loaiMH) => DropdownMenuItem<String>(
    child: Text(loaiMH),
    value: loaiMH,
  )).toList(),
  onChanged: (value) {
    mh.loaiMH=value;
  },
  validator: (value) => value ==null? "Chưa chọn loại mặt hàng" : null,
  decoration: InputDecoration(
    labelText: 'Loại mặt hàng',
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(10))
  ),
),
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 112

## RaiseButton

```


RaisedButton(
  child: Text('OK'),
  color: Colors.white,
  onPressed: () {
    if(_formState.currentState.validate())
    {
      _formState.currentState.save();
      _showAlertDialog(context);
    }
  },),

void _showAlertDialog(BuildContext context){
  AlertDialog alertDialog = AlertDialog(
    title: Text('Xác nhận'),
    content: Text("Bạn nhập mặt hàng: ${mh.tenMH}\n"
      "Thuộc loại mặt hàng: ${mh.loaiMH}\n"
      "Với số lượng: ${mh.soLuong}"),
    actions: [
      FlatButton(
        onPressed: () => Navigator.pop(context),
        child: Text('Ok'))
    ],
  );
  showDialog(
    context: context,
    builder:(context) => alertDialog,
  );
}

```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 113

## 7. Navigation và Routing



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 114

## Navigate tới màn hình mới và quay ngược lại

- Tạo hai route
  - Trong Flutter thuật ngữ Route ám chỉ một màn hình (screen) hoặc trang (page)
- Điều hướng tới route thứ 2 sử dụng Navigator.push().
- Quay lại route thứ nhất bằng cách sử dụng Navigator.pop().

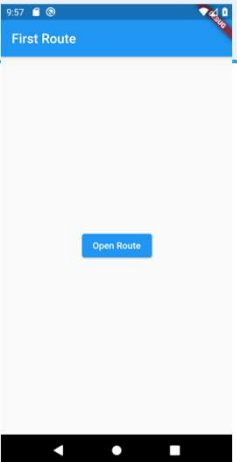
Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 115

## Route thứ nhất

```

class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("First Route")),
      body: Center(
        child: ElevatedButton(
          child: Text('Open Route'),
          onPressed: () => Navigator.push(context,
            MaterialPageRoute(
              builder: (context) => SecondRoute(),)),
        ),
      ),
    );
  }
}

```

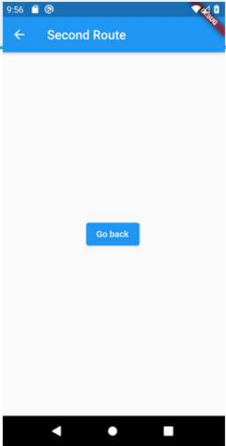


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT 116



## Route thứ hai

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Second Route')),
      body: Center(
        child: ElevatedButton(
          child: Text('Go back'),
          onPressed: () => Navigator.pop(context),
        ),
      ),
    );
  }
}
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

117

## Navigate with named routes

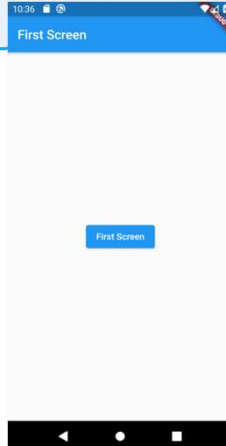
1. Tạo hai màn hình.
2. Định nghĩa các routes.
3. Navigate tới second màn hình hai bằng cách sd Navigator.pushNamed().
4. Trở về màn hình thứ nhất bằng các sd Navigator.pop().

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

118

## First Screen

```
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('First Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('First Screen'),
          onPressed: () =>
            Navigator.pushNamed(context, '/second'),
        ),
      ),
    );
  }
}
```

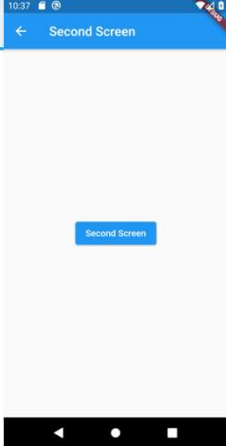


Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

119

## Second Screen

```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Second Screen')),
      body: Center(
        child: ElevatedButton(
          child: Text('Second Screen'),
          onPressed: () => Navigator.pop(context),
        ),
      ),
    );
  }
}
```



Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

120

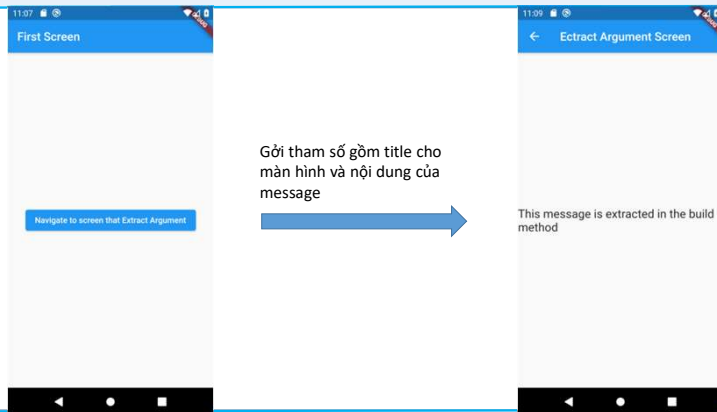
## Định nghĩa các route

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => FirstScreen(),
        '/second': (context) => SecondScreen()
      },
    );
  }
}
```

## Truyền tham số cho named route

- Định nghĩa các đối số cần truyền.
- Tạo một widget sử dụng các đối số: Trích xuất các đối số bằng `ModalRoute.of()`
- Đăng ký widget trong bảng route
- Điều hướng tới widget: sử dụng `Navigator.pushNamed()`

## Ví dụ



## Định nghĩa tham số cần truyền

- Định nghĩa lớp dữ liệu biểu diễn tham số cần truyền.
- Ví dụ:

```
class ScreenArguments{
  String title;
  String message;

  ScreenArguments(this.title, this.message);
}
```

## Tạo widget sử dụng các đối số được truyền vào route

- Để truy cập các đối số, sử dụng phương thức `ModalRoute.of()`. Phương thức này trả về route hiện tại với các tham số của nó.

```
class ExtractArgumentScreen extends StatelessWidget {
  static const routeName = '/extractArguments';
  @override
  Widget build(BuildContext context) {
    final ScreenArguments args =
      ModalRoute.of(context).settings.arguments;
    return Scaffold(
      appBar: AppBar(
        title: Text(args.title),
      ),
      body: Center(
        child: Text(args.message,
          style: TextStyle(fontSize: 20)),
      ),
    );
  }
}
```

## Đăng ký widget với bảng route

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => ProviderArgumentScreen(),
        ExtractArgumentScreen.routeName: (context) => ExtractArgumentScreen(),
      },
    );
  }
}
```

## Điều hướng tới widget

- Cung cấp đối số cho route thông qua thuộc tính `arguments`

```
class ProviderArgumentScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("First Screen"),
      ),
      body: Center(
        child: ElevatedButton(
          child: Text('Navigate to screen that Extract Argument'),
          onPressed: () => Navigator.pushNamed(context,
            ExtractArgumentScreen.routeName,
            arguments: ScreenArguments(
              'Extract Argument Screen',
              "This message is extracted in the build method"
            ),
          ),
        ),
      ),
    );
  }
}
```

Đối số được  
truyền vào route

## Dữ liệu trả về từ một màn hình

- Sử dụng `navigator.pop()`

```
@optionalTypeArgs
void pop<T extends Object>(
  BuildContext context,
  [T result]
)
```

### Ví dụ:

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

129

```
class SelectionButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () => _navAndDispSelection(context),
      child: Text('Pick an Option, any option!'));
  }
  Future<void> _navAndDispSelection(BuildContext context) async {
    final result = await Navigator.push(context,
      MaterialPageRoute(builder: (context)
        => SelectionPage(),));
    Scaffold.of(context)
      ..removeCurrentSnackBar()
      ..showSnackBar(
        SnackBar(content: Text('$result')));
  }
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

130

```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      _getButtonOption(context, 'Yep'),
      _getButtonOption(context, 'Nope')
    ],
  ),
),

Widget _getButtonOption(BuildContext context, String s) {
  return Padding(
    padding: EdgeInsets.all(8),
    child: ElevatedButton(
      child: Text(s),
      onPressed: () => Navigator.pop(context, s),
    ),
  );
}
```

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

131

### Gửi dữ liệu tới một màn hình mới

- Sử dụng `Navigator.push`
- Ví dụ

Huỳnh Tuấn Anh - Khoa CNTT, ĐHNT

132

## Định nghĩa tham số

```
class Todo{
  String title;
  String description;

  Todo(this.title, this.description);
}

List<Todo> getListTodo() {
  List<Todo> todos = List.generate(20, (index) => Todo(
    "Todo ${index + 1}",
    "Đây là Todo thứ ${index + 1}"
  ));
  return todos;
}
```

## Todo Screen

```
class TodoScreen extends StatelessWidget {
  final List<Todo> todos=getListTodo();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('List todo')),
      body: ListView.builder(
        itemCount: todos.length,
        itemBuilder: (context, index) => Card(
          child: ListTile(
            title: Text(todos[index].title),
            leading: Icon(Icons.work, color: Colors.blue),
            onTap: () => Navigator.push(context,
              MaterialPageRoute(
                builder: (context) => DetailScreen(todo: todos[index]),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Dữ liệu cần gọi sang màn hình chi tiết

## Detail Screen

```
class DetailScreen extends StatelessWidget {
  Todo todo;
  DetailScreen({Key key, this.todo}): super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Todo')),
      body: Column( mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Center(
            child: Text(todo.title,
              style: TextStyle(fontSize: 25, fontWeight: FontWeight.bold,
                color: Colors.blue),
            ),
            padding: const EdgeInsets.all(8.0),
            child: Text(todo.description, style: TextStyle(fontSize: 18,)),
          ),
        ],
      ),
    );
  }
}
```

Dữ liệu của màn hình 2