**Widget trong Flutter**

Như đã nói trong các bài học trước **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 như là các view ở trong **Android**.

Ở trong bài trước thì chúng ta đã tạo được các **widget** đơn giản, trong bài học này chúng ta sẽ tìm hiểu kĩ hơn về cách tạo widget, cách thức hoạt động và các loại widget được hỗ trợ bởi **Flutter**.

Trong ứng dụng Hello World, chúng ta đã tạo một widget tên là *MyHomePage*

class MyHomePage extends StatelessWidget {

MyHomePage({Key key, this.title}) : super(key: key);

final String title;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text(this.title), ),

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

);

}

}

Chúng ta sẽ tìm hiểu kỹ hơn về cách tạo widget này:

**Đầu tiên** chúng ta thấy rằng *MyHomePage*được kết thừa từ một widget khác là*StatelessWidget.*Thông thường trong Flutter để tạo một widget mới (*widget của người dùng,* ko phải widget mặc định) ta cần kế thừa một trong hai class là ***StatelessWidget*** *và****StateFullWidget****.*Để hiểu hơn thì chúng ta sẽ nói rõ trong bài Sate, nhưng chúng ta có thể hiêu đơn giản *StatelessWidget* là một widget không có trang thái nào, nó chỉ nhận dữ liệu và hiển thị một cách thụ động, không nhận bất cứ event nào.

*StatelessWidget*chỉ yêu cầu implement duy nhất một phương thức*build. P*hương thức này lấy thông tin để dựng các widget thông qua tham số *BuildContext* và trả về **widget** mà nó tạo ra.

Như đoạn code ở trên, ta có thể thấy hàm *build* sử dụng thuộc tính *title* từ hàm khởi tạo để hiển thị têu đề cho ứng dụng. Còn tham số *Key* được dùng để định dang cho widget.

Chúng ta thấy hàm *build* lại gọi tới một widget khác là *Scaffold*. Widget này đóng vai trò như một phần nền để bố trí các thành phần khác theo phong cách Material Design, tương tự như *DrawerLayout* và *CoordinatorLayout* trong Android.

Cuối cùng ta thấy có một widget là *Center* có nhiệm vụ bố trí *Text* ở giữa màn hình.

Để hiểu hơn mối quan hệ giữa các widget chúng ta tham khảo sơ đồ dưới đây:

A picture containing text, screenshot, font, diagram

Description automatically generated

Tổng quan về các loại Widget trong Flutter

Trong **Flutter** tất các widget được phân loại dựa trên chức năng thành 4 nhóm sau:

* Các widget giao diện đặc thù theo từng nền tảng -*Platform widgets*
* Các widget hỗ trợ bố trí giao diện - *Layout widgets*
* Các widget quản lý trạng thái - *State maintenance widgets*
* Các widget cơ bản độc lập với nền tảng - Platform independent / *basic widgets*

Chúng ta sẽ tìm hiểu kĩ từng loại widget dưới đây:

Platform specific widgets

Đây là các widget dành riêng cho từng nên tảng Android hay IOS

Các widget dành riêng cho Android được thiết kết theo *Material design guideline* cho Android OS nên được gọi là  ***Material*** *widgets*.

Các widget dành riêng cho iOS được thiết kế theo *Human Interface Guidelines* bởi Apple và được gọi là ***Cupertino*** widgets

Một số material widgets phổ biến nhất cho Android:

* Scaffold
* AppBar
* BottomNavigationBar
* TabBar
* TabBarView
* ListTile
* RaisedButton
* FloatingActionButton
* FlatButton
* IconButton
* DropdownButton
* PopupMenuButton
* ButtonBar
* TextField
* Checkbox
* Radio
* Switch
* Slider
* Date & Time Pickers
* SimpleDialog
* AlertDialog

Một số *Cupertino* widgets phổ biến nhất cho IOS

* CupertinoButton
* CupertinoPicker
* CupertinoDatePicker
* CupertinoTimerPicker
* CupertinoNavigationBar
* CupertinoTabBar
* CupertinoTabScaffold
* CupertinoTabView
* CupertinoTextField
* CupertinoDialog
* CupertinoDialogAction
* CupertinoFullscreenDialogTransition
* CupertinoPageScaffold
* CupertinoPageTransition
* CupertinoActionSheet
* CupertinoActivityIndicator
* CupertinoAlertDialog
* CupertinoPopupSurface
* CupertinoSlider

Layout widgets

Trong Flutter, một widget có thể được tạo thành từ một hoặc nhiều widget khác. Việc kết hợp nhiều widget thành một widget được thực hiện thông qua các layout widget. Ví dụ, các widget con có thể được căn giữa thông *Center* widget.

Một số layout widgets phổ biến:

1. **Container** − Một hình chữ nhật được thiết kế sử dụng *BoxDecoration* widgets với background (nền), border (đường viền) và shadow (bóng đổ).
2. **Center** − Căn giữa các widget con.
3. **Row** − Sắp xếp các widget con theo hàng ngang (horizontal direction).
4. **Column** − Sắp xếp các widget con theo hàng dọc (vertical direction).
5. **Stack** − Sắp xếp các widget con lên trên cùng

Chúng ta sẽ tim hiểu kỹ hơn trong bài *Giới thiệu layout widget* ở các bài sau.

State maintenance widgets

Trong **Flutter**, tất cả các widget đều kế thừa từ *StatelessWidget* hoặc *StatefulWidget*.

Widget kế thừa từ *StatelessWidget* sẽ không có bất kì trạng thái nào nhưng nó có thể bao gồm widget được kết thừa từ *StatefulWidget*. Bản chất sự linh hoạt của ứng dụng là thông qua hành vi tương tác của các widget và sự thay đổi trạng thái của chúng. Ví dụ khi chạm vào một nút tăng giảm của bộ đếm, nó sẽ làm tăng hoặc giảm trạng thái của bộ đếm trong widget và cơ chế *reactive nature* (tự phản ứng) sẽ tự động thay đổi lại giao diện của widget theo trạng thái mới.

Chúng ta sẽ tìm hiểu kỹ hơn về khái niệm *StatefulWidget* trong bài học về *State management*

Platform independent / basic widgets

*Flutter* cung cấp một số lương lớn các widget cơ bản để tạo các giao diện người dùng từ đơn giản đến phức tạp độc lập với nền tảng hệ điều hành. Chúng ta sẽ tìm hiểu một số widget cơ bản dưới đây:

**Text**

*Text* widget được sử dụng để hiển thị một đoạn văn bản. Chúng ta có thể định dạng văn bản thông qua thuộc tính *style* và*TextStyle* class. Ví dụ:

Text('Hello World!', style: TextStyle(fontWeight: FontWeight.bold))

*Text* widget có một hàm constructor riêng, *Text.rich*, sử dụng một *TextSpan* để mô tả các định dang. *TextSpan* widget có tính chất đệ quy và nó có thể bao gồm các *TextSpan* khác. Ví dụ:

Text.rich(

TextSpan(

children: [

TextSpan(text: "Hello ", style:

TextStyle(fontStyle: FontStyle.italic)),

TextSpan(text: "World", style:

TextStyle(fontWeight: FontWeight.bold)),

],

),

)

Một số thuộc tính cơ bản của Text widget

* **maxLines, int** − Số lượng dòng tối đa
* **overflow, TextOverFlow** − Xỷ lý việc tràn văn bản sử dụng *TextOverFlow* class
* **style, TextStyle** − Mô tả định dang văn bản thông *TextStyle* class
* **textAlign, TextAlign** − Căn lề văn bản: right, left, justify ,... sử dụng *TextAlign* class
* **textDirection, TextDirection** − Quy đinh chiều của văn bản left-to-right hoặc right-to-left

**Image**

*Image* widget được sử dụng để hiển thị hình ảnh trong ứng dụng. *Image* widget cung cấp các phương thức khởi tạo khác nhau để load hình ảnh từ các nguồn khác nhau:

* **Image** − Hình ảnh thông thường sử dụng *ImageProvider*
* **Image.asset** − Load hình ảnh từ flutter project’s assets
* **Image.file** − Load hình ảnh từ system folder
* **Image.memory** − Load hình ảnh từ memory
* **Image.Network** − Load hình ảnh từ mạng network

Lựa chọn đơn giản nhất để hiển thị hình ảnh trong *Flutter* là đưa hình ảnh vào thư mục assets của ứng dụng rồi load vào widget khi cần:

* Tạo một thư mục assets ở trong project và copy file ảnh lưu vào.

A screenshot of a computer

Description automatically generated

Right click 🡪 Myapp 🡪 New 🡪 Directory

* Mô tả assets ở trong file *pubspec.yaml* như sau

flutter:

assets:

- assets/smiley.png

A screenshot of a computer

Description automatically generated

Nhớ đúng cú pháp trên nhé

* Sau đó, load và hiển thị ảnh trong Ứng dụng bằng widget.

Image.asset('assets/smiley.png')

* Thay thế code MyHomePage widget trong ứng dụng hello world như sau:

class MyHomePage extends StatelessWidget {

MyHomePage({Key key, this.title}) : super(key: key);

final String title;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar( title: Text(this.title), ),

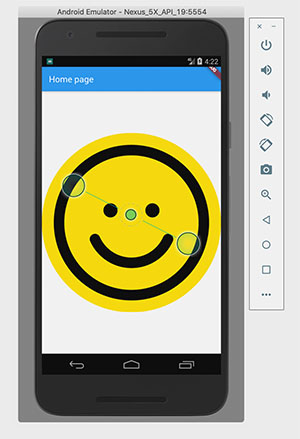
body: Center( child: Image.asset("assets/smiley.png")),

);

}

}

Chạy thử ứng dụng hình ảnh sẽ được load lên như sau:



Một số thuộc tính cơ bản của *Image* widget:

* **image, ImageProvider** − Kích thước ảnh
* **width, double** − Độ rộng của ảnh
* **height, double** − Độ cao của ảnh
* **alignment, AlignmentGeometry** − Căn lề

**Icon**

*Icon* widget hiển thị hình ảnh các icon cơ bản trong *IconData* class.

Icon(Icons.email)

Chúng ta sửa lại code MyHomePage widget như sau:

class MyHomePage extends StatelessWidget {

MyHomePage({Key key, this.title}) : super(key: key);

final String title;

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text(this.title),),

body: Center( child: Icon(Icons.email)),

);

}

}

Chạy thử trên máy ảo

