1. AnimatedWidget:

AnimatedWidget là một widget rất hữu ích trong Flutter để tạo các hiệu ứng hoạt hình mà không cần phải làm việc trực tiếp với AnimationController. AnimatedWidget là một lớp trừu tượng mà bạn có thể kế thừa và triển khai phương thức build() để trả về giao diện người dùng cụ thể, phụ thuộc vào giá trị của Animation. Khi giá trị Animation thay đổi, Flutter sẽ tự động gọi hàm build để cập nhật giao diện.

Ví dụ:

class MyAnimatedWidget extends AnimatedWidget {

MyAnimatedWidget({Key? key, required Animation<double> animation})

: super(key: key, listenable: animation);

@override

Widget build(BuildContext context) {

final Animation<double> animation = listenable as Animation<double>;

return Opacity(

opacity: animation.value,

child: Text(

'Fading Text',

style: TextStyle(fontSize: 20),

),

);

}

}

1. AnimatedBuilder:

AnimatedBuilder cũng giống như AnimatedWidget, tuy nhiên nó không yêu cầu bạn phải kế thừa lớp trừu tượng. Thay vào đó, bạn chỉ cần truyền Animation vào AnimatedBuilder thông qua hàm builder, và AnimatedBuilder sẽ tự động gọi builder với giá trị Animation mới khi Animation thay đổi.

Ví dụ:

AnimationController controller = AnimationController(

vsync: this,

duration: Duration(seconds: 1),

);

Animation<double> animation = Tween(begin: 0.0, end: 1.0).animate(controller);

AnimatedBuilder(

animation: animation,

builder: (context, child) {

return Opacity(

opacity: animation.value,

child: Text(

'Fading Text',

style: TextStyle(fontSize: 20),

),

);

},

)

1. AnimationController:

AnimationController là một lớp điều khiển hoạt hình trong Flutter. Nó giúp bạn tạo và quản lý giá trị animation trong một khoảng thời gian nhất định. Bạn có thể điều chỉnh thời gian, điều khiển việc lặp lại hoặc dừng hoạt hình với AnimationController.

Ví dụ:

AnimationController controller = AnimationController(

vsync: this,

duration: Duration(seconds: 1),

);

controller.forward(); // Bắt đầu hoạt hình

controller.addListener(() {

// Cập nhật giao diện người dùng dựa trên giá trị của AnimationController

setState(() {});

});

1. CurvedAnimation:

CurvedAnimation là một lớp để tạo các hiệu ứng hoạt hình có độ chậm hoặc nhanh dần dựa trên hàm cong. Bạn có thể cung cấp một AnimationController và một đối tượng Curve để điều chỉnh tốc độ hoạt hình.

Ví dụ:

AnimationController controller = AnimationController(

vsync: this,

duration: Duration(seconds: 1),

);

CurvedAnimation curve = CurvedAnimation(

parent: controller,

curve: Curves.easeInOut,

);

Animation<double> animation = Tween(begin: 0.0, end: 1.0).animate(curve);

1. Hero:

Hero là một widget trong Flutter cho phép bạn tạo các hiệu ứng chuyển đổi mượt mà giữa hai widget khi chuyển đổi giữa các màn hình khác nhau. Widget Hero có thể giúp tạo hiệu ứng đẹp mắt và chuyển đổi trực quan giữa các trang.

Ví dụ:

class FirstPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('First Page')),

body: GestureDetector(

onTap: () {

Navigator.push(

context,

MaterialPageRoute(

builder: (context) => SecondPage(),

),

);

},

child: Hero(

tag: 'heroTag',

child: Image.asset('assets/image.png'),

),

),

);

}

}

class SecondPage extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(title: Text('Second Page')),

body: Center(

child: Hero(

tag: 'heroTag',

child: Image.asset('assets/image.png'),

),

),

);

}

}

1. Opacity:

Opacity là một widget trong Flutter cho phép bạn điều chỉnh độ mờ của widget con trong khoảng từ 0.0 (hoàn toàn mờ) đến 1.0 (hoàn toàn rõ nét).

Ví dụ:

Opacity(

opacity: 0.5, // Điều chỉnh độ mờ

child: Container(

color: Colors.blue,

width: 100,

height: 100,

),

)