Họ và Tên: Võ Xuân Trường

MSSV: 0850080109

Lớp 08\_ĐH\_THMT

Nội dung Animations:

* AnimatedWidget: AnimatedWidget là một widget trừu tượng trong Flutter được sử dụng để xây dựng các widget phản hồi đến các giá trị động thay đổi trong Animation. Khi giá trị của Animation thay đổi, AnimatedWidget sẽ tự động tái xây dựng và cập nhật lại UI.

VD:

import 'dart:math' as math;

import 'package:flutter/material.dart';

/// Flutter code sample for [AnimatedWidget].

void main() => runApp(const AnimatedWidgetExampleApp());

class AnimatedWidgetExampleApp extends StatelessWidget {

const AnimatedWidgetExampleApp({super.key});

@override

Widget build(BuildContext context) {

return const MaterialApp(

home: AnimatedWidgetExample(),

);

}

}

class SpinningContainer extends AnimatedWidget {

const SpinningContainer({

super.key,

required AnimationController controller,

}) : super(listenable: controller);

Animation<double> get \_progress => listenable as Animation<double>;

@override

Widget build(BuildContext context) {

return Transform.rotate(

angle: \_progress.value \* 2.0 \* math.pi,

child: Container(width: 200.0, height: 200.0, color: Colors.green),

);

}

}

class AnimatedWidgetExample extends StatefulWidget {

const AnimatedWidgetExample({super.key});

@override

State<AnimatedWidgetExample> createState() => \_AnimatedWidgetExampleState();

}

/// [AnimationController]s can be created with `vsync: this` because of

/// [TickerProviderStateMixin].

class \_AnimatedWidgetExampleState extends State<AnimatedWidgetExample>

with TickerProviderStateMixin {

late final AnimationController \_controller = AnimationController(

duration: const Duration(seconds: 10),

vsync: this,

)..repeat();

@override

void dispose() {

\_controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return SpinningContainer(controller: \_controller);

}

}

* AnimatedBuilder: AnimatedBuilder cũng giống như AnimatedWidget, nhưng nó không trực tiếp kế thừa từ Widget. Nó là một widget builder, cho phép bạn xây dựng một widget tùy chỉnh dựa trên các giá trị thay đổi của Animation.

VD:

import 'dart:math' as math;

import 'package:flutter/material.dart';

/// Flutter code sample for [AnimatedBuilder].

void main() => runApp(const AnimatedBuilderExampleApp());

class AnimatedBuilderExampleApp extends StatelessWidget {

const AnimatedBuilderExampleApp({super.key});

@override

Widget build(BuildContext context) {

return const MaterialApp(

home: AnimatedBuilderExample(),

);

}

}

class AnimatedBuilderExample extends StatefulWidget {

const AnimatedBuilderExample({super.key});

@override

State<AnimatedBuilderExample> createState() => \_AnimatedBuilderExampleState();

}

/// AnimationControllers can be created with `vsync: this` because of

/// TickerProviderStateMixin.

class \_AnimatedBuilderExampleState extends State<AnimatedBuilderExample>

with TickerProviderStateMixin {

late final AnimationController \_controller = AnimationController(

duration: const Duration(seconds: 10),

vsync: this,

)..repeat();

@override

void dispose() {

\_controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return AnimatedBuilder(

animation: \_controller,

child: Container(

width: 200.0,

height: 200.0,

color: Colors.green,

child: const Center(

child: Text('Whee!'),

),

),

builder: (BuildContext context, Widget? child) {

return Transform.rotate(

angle: \_controller.value \* 2.0 \* math.pi,

child: child,

);

},

);

}

}

* AnimationController: AnimationController là một lớp điều khiển animation trong Flutter. Nó giúp quản lý thời gian và các giá trị của animation. Bạn có thể điều khiển việc chạy, dừng, chuyển hướng, và đảo ngược animation thông qua AnimationController.

VD:

* CuvedAnimation: CurvedAnimation là một dạng animation decorator (trang trí) trong Flutter, được sử dụng để áp dụng một hiệu ứng cong (curve) vào animation. Hiệu ứng cong có thể là tăng tốc (accelerate), giảm tốc (decelerate), hoặc kết hợp cả hai.
* Hero: Hero là một widget trong Flutter được sử dụng để tạo hiệu ứng chuyển đổi mượt mà giữa hai màn hình khi chúng chuyển từ màn hình này sang màn hình khác. Hero widget thường được sử dụng khi bạn muốn các phần tử trong hai màn hình trông như nhau và chuyển đổi giữa chúng có hiệu ứng liên tục.
* Opacity: Opacity là một widget trong Flutter được sử dụng để điều chỉnh độ trong suốt của các widget con của nó. Bằng cách thay đổi độ trong suốt, bạn có thể làm cho các widget con hiển thị hoặc ẩn đi một cách mượt mà.