**Animations**

Nguyễn Thị Ngọc Hiền

MSSV: 0850080100

Lớp: 08\_DH\_CNPM

**1. AnimatedWidget:** AnimatedWidget là một lớp trừu tượng trong Flutter, được sử dụng để xây dựng các widget tương tác có hoạt hình. Nó cung cấp một cách thuận tiện để làm việc với các widget dựa trên giá trị animation. Khi giá trị animation thay đổi, AnimatedWidget sẽ tự động tái xây dựng và cập nhật giao diện người dùng.

**Ví dụ:** Sử dụng AnimatedWidget để tạo một hộp di chuyển từ vị trí ban đầu đến vị trí mới.

|  |
| --- |
| import 'package:flutter/material.dart';  class AnimatedBox extends AnimatedWidget {  AnimatedBox({required this.animation})  : super(listenable: animation);  final Animation<double> animation;  @override  Widget build(BuildContext context) {  return Transform.translate(  offset: Offset(animation.value, 0),  child: Container(  width: 100,  height: 100,  color: Colors.blue,  ),  );  }  }  void main() {  runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(title: Text('AnimatedWidget Example')),  body: Center(  child: MyAnimation(),  ),  ),  ));  }  class MyAnimation extends StatefulWidget {  @override  \_MyAnimationState createState() => \_MyAnimationState();  }  class \_MyAnimationState extends State<MyAnimation>  with SingleTickerProviderStateMixin {  late AnimationController \_controller;  late Animation<double> \_animation;  @override  void initState() {  super.initState();  \_controller = AnimationController(  vsync: this,  duration: Duration(seconds: 2),  );  \_animation = Tween<double>(begin: 0, end: 200).animate(\_controller);  \_controller.forward();  }  @override  void dispose() {  \_controller.dispose();  super.dispose();  }  @override  Widget build(BuildContext context) {  return AnimatedBox(animation: \_animation);  }  } |

**2. AnimatedBuilder:** AnimatedBuilder cũng giống như AnimatedWidget nhưng sử dụng một cách khác để xây dựng widget với animation. Nó cho phép bạn sử dụng một hàm builder để xây dựng widget một cách tùy chỉnh, truyền vào Animation và Widget con như các tham số.

**Ví dụ:** Sử dụng AnimatedBuilder để tạo một hộp có kích thước thay đổi theo giá trị của animation.

|  |
| --- |
| import 'package:flutter/material.dart';  class AnimatedBox extends StatefulWidget {  @override  \_AnimatedBoxState createState() => \_AnimatedBoxState();  }  class \_AnimatedBoxState extends State<AnimatedBox>  with SingleTickerProviderStateMixin {  late AnimationController \_controller;  late Animation<double> \_animation;  @override  void initState() {  super.initState();  \_controller = AnimationController(  vsync: this,  duration: Duration(seconds: 2),  )..repeat(reverse: true);  \_animation = Tween<double>(begin: 100, end: 200).animate(\_controller);  }  @override  void dispose() {  \_controller.dispose();  super.dispose();  }  @override  Widget build(BuildContext context) {  return Center(  child: AnimatedBuilder(  animation: \_animation,  builder: (BuildContext context, Widget? child) {  return Container(  width: \_animation.value,  height: \_animation.value,  color: Colors.blue,  );  },  ),  );  }  }  void main() {  runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(title: Text('AnimatedBuilder Example')),  body: AnimatedBox(),  ),  ));  } |

**3. AnimationController:** AnimationController là một lớp trong Flutter giúp quản lý các Animation. Nó cho phép bạn điều khiển việc thực hiện animation, như bắt đầu, dừng, tiếp tục, và hủy bỏ animation. Nó cũng quản lý thời gian và giá trị của Animation.

**Ví dụ:** Sử dụng AnimationController để tạo một hộp có thể được kéo vào bên phải khi nhấn vào nút "Start" và kéo lại bên trái khi nhấn vào nút "Reset".

|  |
| --- |
| import 'package:flutter/material.dart';  class AnimatedBox extends StatefulWidget {  @override  \_AnimatedBoxState createState() => \_AnimatedBoxState();  }  class \_AnimatedBoxState extends State<AnimatedBox>  with SingleTickerProviderStateMixin {  late AnimationController \_controller;  late Animation<double> \_animation;  @override  void initState() {  super.initState();  \_controller = AnimationController(  vsync: this,  duration: Duration(seconds: 2),  );  \_animation = Tween<double>(begin: 0, end: 200).animate(\_controller);  }  @override  void dispose() {  \_controller.dispose();  super.dispose();  }  void \_startAnimation() {  \_controller.forward();  }  void \_resetAnimation() {  \_controller.reverse();  }  @override  Widget build(BuildContext context) {  return Center(  child: Column(  mainAxisAlignment: MainAxisAlignment.center,  children: [  AnimatedBuilder(  animation: \_animation,  builder: (BuildContext context, Widget? child) {  return Container(  width: \_animation.value,  height: 100,  color: Colors.blue,  );  },  ),  SizedBox(height: 20),  ElevatedButton(  onPressed: \_startAnimation,  child: Text('Start'),  ),  ElevatedButton(  onPressed: \_resetAnimation,  child: Text('Reset'),  ),  ],  ),  );  }  }  void main() {  runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(title: Text('AnimationController Example')),  body: AnimatedBox(),  ),  ));  } |

**4. CurvedAnimation:** CurvedAnimation là một lớp dùng để cung cấp một hiệu ứng gì đó khi diễn ra animation, không chỉ sử dụng tuyến tính mà có thể thay đổi tốc độ animation dựa trên một hàm góc cong (curve function). Điều này giúp làm cho animation trở nên mượt mà và tự nhiên hơn.

**Ví dụ:** Sử dụng CurvedAnimation để tạo một hộp di chuyển từ vị trí ban đầu đến vị trí mới với hiệu ứng curve tự nhiên.

|  |
| --- |
| import 'package:flutter/material.dart';  class AnimatedBox extends StatefulWidget {  @override  \_AnimatedBoxState createState() => \_AnimatedBoxState();  }  class \_AnimatedBoxState extends State<AnimatedBox>  with SingleTickerProviderStateMixin {  late AnimationController \_controller;  late Animation<double> \_animation;  @override  void initState() {  super.initState();  \_controller = AnimationController(  vsync: this,  duration: Duration(seconds: 2),  );  // Tạo CurvedAnimation với curve tự nhiên  \_animation = CurvedAnimation(parent: \_controller, curve: Curves.easeInOut);  \_controller.forward();  }  @override  void dispose() {  \_controller.dispose();  super.dispose();  }  @override  Widget build(BuildContext context) {  return Center(  child: AnimatedBuilder(  animation: \_animation,  builder: (BuildContext context, Widget? child) {  return Transform.translate(  offset: Offset(\_animation.value \* 200, 0),  child: Container(  width: 100,  height: 100,  color: Colors.blue,  ),  );  },  ),  );  }  }  void main() {  runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(title: Text('CurvedAnimation Example')),  body: AnimatedBox(),  ),  ));  } |

**5. Hero:** Hero là một widget trong Flutter cho phép bạn thực hiện hiệu ứng chuyển đổi đẹp giữa các màn hình hoặc các widget trong cùng một màn hình. Nó cho phép bạn chia sẻ một widget giữa các màn hình và tự động thực hiện animation khi widget được chuyển từ màn hình này sang màn hình khác.

**Ví dụ:** Sử dụng Hero để tạo hiệu ứng chuyển đổi mượt mà giữa hai màn hình chứa cùng một hình ảnh.

|  |
| --- |
| import 'package:flutter/material.dart';  class Screen1 extends StatelessWidget {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: Text('Screen 1')),  body: Center(  child: GestureDetector(  onTap: () {  Navigator.push(  context,  MaterialPageRoute(  builder: (\_) => Screen2(),  ),  );  },  child: Hero(  tag: 'imageHero',  child: Image.network(  'https://via.placeholder.com/200',  width: 100,  height: 100,  ),  ),  ),  ),  );  }  }  class Screen2 extends StatelessWidget {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: Text('Screen 2')),  body: Center(  child: GestureDetector(  onTap: () {  Navigator.pop(context);  },  child: Hero(  tag: 'imageHero',  child: Image.network(  'https://via.placeholder.com/400',  width: 200,  height: 200,  ),  ),  ),  ),  );  }  }  void main() {  runApp(MaterialApp(  home: Screen1(),  ));  } |

**6. Opacity:** Opacity là một widget trong Flutter cho phép bạn điều chỉnh độ trong suốt của một widget con. Bằng cách sử dụng Opacity, bạn có thể làm cho một widget trở nên mờ hoặc trong suốt một cách dễ dàng.

**Ví dụ**: Sử dụng Opacity để làm mờ một hình ảnh sau khi nhấn vào nút "Fade".

|  |
| --- |
| import 'package:flutter/material.dart';  class FadeImage extends StatefulWidget {  @override  \_FadeImageState createState() => \_FadeImageState();  }  class \_FadeImageState extends State<FadeImage>  with SingleTickerProviderStateMixin {  bool \_isFaded = false;  late AnimationController \_controller;  @override  void initState() {  super.initState();  \_controller = AnimationController(  vsync: this,  duration: Duration(seconds: 1),  );  }  @override  void dispose() {  \_controller.dispose();  super.dispose();  }  void \_toggleFade() {  if (\_isFaded) {  \_controller.reverse();  } else {  \_controller.forward();  }  \_isFaded = !\_isFaded;  }  @override  Widget build(BuildContext context) {  return Column(  mainAxisAlignment: MainAxisAlignment.center,  children: [  AnimatedBuilder(  animation: \_controller,  builder: (BuildContext context, Widget? child) {  return Opacity(  opacity: \_controller.value,  child: Image.network(  'https://via.placeholder.com/200',  width: 200,  height: 200,  ),  );  },  ),  SizedBox(height: 20),  ElevatedButton(  onPressed: \_toggleFade,  child: Text(\_isFaded ? 'Unfade' : 'Fade'),  ),  ],  );  }  }  void main() {  runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(title: Text('Opacity Example')),  body: Center(  child: FadeImage(),  ),  ),  ));  } |