ICT4153

Mobile Application Development

Practical 08

Flutter Designs and Animations

Objective

To explore the basics of Flutter animations and create a few examples to help you understand how to implement animations in your Flutter applications.

Basic Concepts

1. Animation Controller

The AnimationController is the core of Flutter animations. It controls the animation's duration, start, stop, and reverse. It also provides the current value of the animation.

2. Tween

A Tween is used to define the range of values that an animation can take. For example, you can use a Tween to animate a widget's opacity from 0.0 to 1.0.

3. Animation

An Animation object holds the current value of the animation and notifies listeners when the value changes.

4. AnimatedBuilder

An AnimatedBuilder is a widget that rebuilds itself whenever the animation value changes.

Example 1: Fade-In Animation

Let's start with a simple fade-in animation where a widget gradually becomes visible.

Step 1: Create a new Flutter project

flutter create flutter_animation_example

cd flutter_animation_example

Step 2: Modify lib/main.dart

Replace the content of lib/main.dart with the following code:

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Animation Example'),
        ),
        body: FadeInAnimation(),
      ),
    );
  }
}

class FadeInAnimation extends StatefulWidget {
  @override
  _FadeInAnimationState createState() => _FadeInAnimationState();
}

class _FadeInAnimationState extends State<FadeInAnimation>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Use 'late' here
  late Animation<double> _animation; // Use 'late' here

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );
    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(_controller);
    _controller.forward();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: FadeTransition(
        opacity: _animation,
        child: Container(
          width: 200.0,
          height: 200.0,
          color: Colors.blue,
          child: Center(
            child: Text(
              'Fade In',
              style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

- AnimationController: We create an AnimationController with a duration of 2 seconds and set vsync to this to sync the animation with the screen refresh rate.
- Tween: We define a Tween that animates the opacity from 0.0 (invisible) to 1.0 (fully visible).
- FadeTransition: We use the FadeTransition widget to apply the animation to the opacity of the container.

Step 3: Run the app

flutter run

You should see a blue container that fades in over 2 seconds.

Example 2: Scale Animation

Next, let's create a scale animation where a widget grows in size.

Step 1: Modify lib/main.dart
Replace the content of lib/main.dart with the following code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Animation Example'),
        ),
        body: ScaleAnimation(),
      ),
    );
  }
}

class ScaleAnimation extends StatefulWidget {
  @override
  _ScaleAnimationState createState() => _ScaleAnimationState();
}

class _ScaleAnimationState extends State<ScaleAnimation>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Use 'late' keyword
  late Animation<double> _animation; // Use 'late' keyword

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.easeInOut,
      ),
    );

    _controller.forward();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: ScaleTransition(
        scale: _animation,
        child: Container(
          width: 200.0,
          height: 200.0,
          color: Colors.green,
          child: Center(
            child: Text(
              'Scale',
              style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

- CurvedAnimation: We use CurvedAnimation to apply an easing curve to the animation, making it start slow, speed up, and then slow down again.

- ScaleTransition: We use the ScaleTransition widget to apply the animation to the scale of the container.

Step 2: Run the app

flutter run

You should see a green container that grows in size over 2 seconds with an easing effect.

Example 3: Rotation Animation

Finally, let's create a rotation animation where a widget rotates around its center.

Step 1: Modify lib/main.dart

Replace the content of lib/main.dart with the following code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flutter Animation Example'),
        ),
        body: RotationAnimation(),
      ),
    );
  }
}

class RotationAnimation extends StatefulWidget {
  @override
  _RotationAnimationState createState() => _RotationAnimationState();
}

class _RotationAnimationState extends State<RotationAnimation>
```

```
  with SingleTickerProviderStateMixin {
 late AnimationController _controller; // Use 'late'
 late Animation<double> _animation; // Use 'late'

 @override
 void initState() {
  super.initState();

  _controller = AnimationController(
   duration: const Duration(seconds: 2),
   vsync: this,
  );

  _animation = Tween<double>(begin: 0.0, end: 1.0).animate(
   CurvedAnimation(
    parent: _controller,
    curve: Curves.easeInOut,
   ),
  );

  _controller.repeat(); // If you want it to run indefinitely
  // Use `_controller.forward();` instead if you want a one-time animation
 }

 @override
 void dispose() {
  _controller.dispose();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return Center(
   child: RotationTransition(
    turns: _animation,
    child: Container(
     width: 200.0,
     height: 200.0,
     color: Colors.orange,
     child: Center(
      child: Text(
       'Rotate',
       style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

- RotationTransition: We use the RotationTransition widget to apply the animation to the rotation of the container.
- Repeat: We use _controller.repeat() to make the animation loop indefinitely.

Step 2: Run the app

flutter run

You should see an orange container that rotates continuously around its center.

Activity

Complete the given activity below.

Instructions

1. Ensure that Flutter and Dart SDKs are installed on your system. Use an IDE like Android Studio or VS Code with Flutter and Dart plugins installed.
2. Create a new Flutter project named flutter_animation_practical.
3. Implement the Following Animations:
    - **Fade-In Animation**
    - **Scale Animation**
    - **Rotation Animation**
4. Create a new animation that combines two or more of the above animations.
5. Experiment with different properties, curves, and durations to customize the animations.

Tasks

Task 1: Fade-In Animation

1. Create a new Flutter project.
2. Implement a fade-in animation where a widget gradually becomes visible over 3 seconds.
3. Use AnimationController, Tween, and FadeTransition to achieve this.
4. Run the app and observe the fade-in effect.

Task 2: Scale Animation

1. Implement a scale animation where a widget grows in size over 2 seconds.
2. Use AnimationController, Tween, CurvedAnimation, and ScaleTransition to achieve this.
3. Apply an easing curve to make the animation start slow, speed up, and then slow down again.
4. Run the app and observe the scale effect.

Task 3: Rotation Animation

1. Implement a rotation animation where a widget rotates around its center continuously.
2. Use AnimationController, Tween, CurvedAnimation, and RotationTransition to achieve this.
3. Make the animation loop indefinitely.
4. Run the app and observe the rotation effect.

Task 4: Combine Animations

1. Create a new animation that combines fade-in and scale animations.
2. Use AnimationController, Tween, CurvedAnimation, FadeTransition, and ScaleTransition to achieve this.
3. Ensure the widget fades in and grows in size simultaneously over 3 seconds.
4. Run the app and observe the combined effect.

Task 5: Customize Animations

1. Customize the animations by changing properties like duration, curves, and values.
2. Experiment with different easing curves (e.g., Curves.bounceIn, Curves.elasticOut).
3. Add a button to start and stop the animations.
4. Run the app and observe the customized effects.

Sample Code

Task 1: Fade-In Animation

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Fade-In Animation'),
        ),
        body: FadeInAnimation(),
      ),
    );
  }
}

class FadeInAnimation extends StatefulWidget {
  @override
  _FadeInAnimationState createState() => _FadeInAnimationState();
}

class _FadeInAnimationState extends State<FadeInAnimation>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Use 'late'
  late Animation<double> _animation; // Use 'late'

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(_controller);

    _controller.forward();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: FadeTransition(
        opacity: _animation,
        child: Container(
          width: 200.0,
          height: 200.0,
          color: Colors.blue,
          child: Center(
            child: Text(
              'Fade In',
              style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

Task 2: Scale Animation

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Scale Animation'),
        ),
        body: ScaleAnimation(),
      ),
    );
  }
}

class ScaleAnimation extends StatefulWidget {
  @override
  _ScaleAnimationState createState() => _ScaleAnimationState();
}
```

```
class _ScaleAnimationState extends State<ScaleAnimation>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Use 'late'
  late Animation<double> _animation; // Use 'late'

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.easeInOut,
      ),
    );

    _controller.forward();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: ScaleTransition(
        scale: _animation,
        child: Container(
          width: 200.0,
          height: 200.0,
          color: Colors.green,
          child: Center(
            child: Text(
              'Scale',
              style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

Task 3: Rotation Animation

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Rotation Animation'),
        ),
        body: RotationAnimation(),
      ),
    );
  }
}

class RotationAnimation extends StatefulWidget {
  @override
  _RotationAnimationState createState() => _RotationAnimationState();
}

class _RotationAnimationState extends State<RotationAnimation>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Fixed null safety
  late Animation<double> _animation; // Fixed null safety

  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.easeInOut,
      ),
    );

    _controller.repeat(); // Runs indefinitely
    // Use `_controller.forward();` instead for a one-time animation
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: RotationTransition(
        turns: _animation,
        child: Container(
          width: 200.0,
          height: 200.0,
          color: Colors.orange,
          child: Center(
            child: Text(
              'Rotate',
              style: TextStyle(color: Colors.white, fontSize: 24.0),
            ),
          ),
        ),
      ),
    );
  }
}
```

Task 4: Combine Animations

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Combine Animations'),
        ),
        body: CombineAnimations(),
      ),
    );
  }
}

class CombineAnimations extends StatefulWidget {
  @override
  _CombineAnimationsState createState() => _CombineAnimationsState();
}

class _CombineAnimationsState extends State<CombineAnimations>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Fixed null safety
  late Animation<double> _fadeAnimation; // Fixed null safety
  late Animation<double> _scaleAnimation; // Fixed null safety

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 3),
      vsync: this,
    );

    _fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(_controller);
    _scaleAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.easeInOut,
      ),
    );

    _controller.forward();
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: FadeTransition(
        opacity: _fadeAnimation,
        child: ScaleTransition(
          scale: _scaleAnimation,
          child: Container(
            width: 200.0,
            height: 200.0,
            color: Colors.purple,
            child: Center(
              child: Text(
                'Combine',
                style: TextStyle(color: Colors.white, fontSize: 24.0),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Task 5: Customize Animations

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Customize Animations'),
        ),
        body: CustomizeAnimations(),
      ),
    );
  }
}

class CustomizeAnimations extends StatefulWidget {
  @override
  _CustomizeAnimationsState createState() => _CustomizeAnimationsState();
}

class _CustomizeAnimationsState extends State<CustomizeAnimations>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller; // Fixed null safety
  late Animation<double> _animation; // Fixed null safety

  @override
  void initState() {
    super.initState();

    _controller = AnimationController(
      duration: const Duration(seconds: 2),
      vsync: this,
    );

    _animation = Tween<double>(begin: 0.0, end: 1.0).animate(
      CurvedAnimation(
        parent: _controller,
        curve: Curves.bounceIn,
      ),
    );

    _controller.repeat(reverse: true);
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ScaleTransition(
            scale: _animation,
            child: Container(
              width: 200.0,
              height: 200.0,
              color: Colors.red,
              child: Center(
                child: Text(
                  'Bounce',
                  style: TextStyle(color: Colors.white, fontSize: 24.0),
                ),
              ),
            ),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              setState(() {
                if (_controller.isAnimating) {
                  _controller.stop();
                } else {
                  _controller.repeat(reverse: true);
                }
              });
            },
            child: Text(_controller.isAnimating ? 'Stop' : 'Start'),
          ),
        ],
      ),
    );
  }
}
```

- **Submit the complete Flutter project with all the implemented animations to the given link in LMS.**