# circular_prograss.dart

## 소스 코드

### 전체 코드

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

class RotatingImageCircularProgressBar extends StatefulWidget {
  final double progress; // 0.0 ~ 1.0
  final double size;
  final String imagePath; // 회전할 이미지 경로

  final Duration progressAnimationDuration;
  final Duration rotationAnimationDuration; // 이미지 회전 애니메이션 한 바퀴 도는 시간

  const RotatingImageCircularProgressBar({
    Key? key,
    required this.progress,
    required this.imagePath,
    this.size = 150.0,
    this.progressAnimationDuration = const Duration(milliseconds: 300),
    this.rotationAnimationDuration = const Duration(seconds: 2),
  }) : super(key: key);

  @override
  _RotatingImageCircularProgressBarState createState() =>
_RotatingImageCircularProgressBarState();
}

class _RotatingImageCircularProgressBarState extends
State<RotatingImageCircularProgressBar>
    with SingleTickerProviderStateMixin { // 이미지 회전 애니메이션을 위해 TickerProvider
필요
  late AnimationController _rotationController;

  @override
  void initState() {
    super.initState();
    _rotationController = AnimationController(
      duration: widget.rotationAnimationDuration,
      vsync: this,
    )..repeat(); // 애니메이션 시작 및 반복
  }

  @override
  void dispose() {
    _rotationController.dispose();
    super.dispose();
  }
```

```dart

  @override
  Widget build(BuildContext context) {
    final double imageActualSize = widget.size * 1;

    return SizedBox(
      width: widget.size,
      height: widget.size,
      child: TweenAnimationBuilder<double>(
        tween: Tween<double>(begin: 0.0, end: widget.progress.clamp(0.0, 1.0)),
        duration: widget.progressAnimationDuration,
        builder: (context, animatedProgress, _) {
          return Stack(
            alignment: Alignment.center,
            children: [
              RotationTransition(
                turns: _rotationController, // AnimationController 사용
                child: Image.asset(
                  widget.imagePath,
                  width: imageActualSize,
                  height: imageActualSize,
                  fit: BoxFit.contain, // 이미지 비율 유지하며 채움
                  // 이미지 로딩 중 오류 발생 시 처리 (선택 사항)
                  errorBuilder: (context, error, stackTrace) {
                    print("Error loading image: $error");
                    return Icon(Icons.broken_image, size: imageActualSize, color:
Colors.grey);
                  },
                ),
              ),
            ],
          );
        },
      ),
    );
  }
}

class GradientCircularProgressPainter extends CustomPainter {
  final double progress; // 0.0 ~ 1.0
  final Color startColor;
  final Color endColor;
  final double strokeWidth;
  final Color? backgroundColor; // 배경 원의 색상 (선택 사항)

  GradientCircularProgressPainter({
    required this.progress,
    required this.startColor,
    required this.endColor,
    this.strokeWidth = 10.0,
    this.backgroundColor,
  });

```

```dart
  @override
  void paint(Canvas canvas, Size size) {
    final Offset center = Offset(size.width / 2, size.height / 2);
    final double radius = math.min(size.width / 2, size.height / 2) - strokeWidth / 2;
    // 그라데이션 영역을 위한 Rect, strokeWidth의 절반을 고려하여 stroke가 중앙에 오도록
    final Rect rect = Rect.fromCircle(center: center, radius: radius);

    // 배경 원 그리기 (선택 사항)
    if (backgroundColor != null) {
      final backgroundPaint = Paint()
        ..color = backgroundColor!
        ..style = PaintingStyle.stroke
        ..strokeWidth = strokeWidth;
      canvas.drawCircle(center, radius, backgroundPaint);
    }

    // 프로그레스 아크(arc) 그리기
    final progressPaint = Paint()
    // SweepGradient를 사용하여 원형 그라데이션 적용
      ..shader = SweepGradient(
        colors: [startColor, endColor],
        startAngle: -math.pi / 2, // 12시 방향에서 시작
        // endAngle은 progress에 따라 동적으로 계산되어야 하지만,
        // SweepGradient 자체는 전체 원에 대한 그라데이션을 정의하고,
        // drawArc에서 그리는 각도로 실제 표시되는 부분을 제어합니다.
        // 하지만 그라데이션이 progress에 따라 변하게 하려면 아래처럼 stops나 transform을 조절해야 할 수 있음.
        // 여기서는 간단하게 전체 그라데이션을 만들고 drawArc로 자르는 방식을 사용.
        // 또는, endAngle을 progress에 맞추고 tileMode를 clamp로 하여 progress 부분만 그라데이션이 적용되게 할 수도 있습니다.
        // 더 정확한 방법은 transform 또는 stops를 조절하는 것입니다.
        // 여기서는 간단히 하기 위해, 전체 원에 그라데이션을 적용하고 drawArc로 해당 부분만 잘라냅니다.
        // 하지만 더 나은 효과를 위해 그라데이션 자체도 progress에 따라 변하도록 할 수 있습니다. (아래 transform 참고)
        stops: [0.0, 1.0], // 그라데이션 색상 정지 지점
        tileMode: TileMode.clamp,
        // 그라데이션 시작점을 12시 방향으로 맞추기 위한 변환
        // transform: GradientRotation(-math.pi / 2), // 이 방법도 가능
      ).createShader(rect) // createShader에는 그라데이션이 적용될 영역(rect)을 전달
      ..style = PaintingStyle.stroke
      ..strokeWidth = strokeWidth
      ..strokeCap = StrokeCap.round; // 선의 끝을 둥글게

    // progress가 0보다 클 때만 그립니다.
    if (progress > 0.0) { // 0일때는 그리지 않도록
      // drawArc의 sweepAngle을 사용하여 progress 만큼만 호를 그립니다.
      // 시작 각도는 -math.pi / 2 (12시 방향)
      // sweepAngle은 progress * 2 * math.pi (전체 원의 각도에 progress 비율을 곱함)
      canvas.drawArc(
        rect, // 아크를 그릴 사각형 영역
        -math.pi / 2, // 시작 각도 (12시 방향)
        math.pi * 2 * progress.clamp(0.0, 1.0), // 진행 각도 (0.0 ~ 1.0 사이 값으로 제한)
```

```
        false, // 호의 중심과 연결하지 않음 (false = 호만 그림, true = 파이 조각)
        progressPaint,
      );
    }
  }

  @override
  bool shouldRepaint(covariant GradientCircularProgressPainter oldDelegate) {
    return progress != oldDelegate.progress ||
        startColor != oldDelegate.startColor ||
        endColor != oldDelegate.endColor ||
        strokeWidth != oldDelegate.strokeWidth ||
        backgroundColor != oldDelegate.backgroundColor;
  }
}
```

- 이미지가 회전하는 애니메이션은 정상 작동
- 하지만 **원형 프로그레스 바는 Stack에 포함되지 않아 화면에 나타나지 않음**

---

## ✅ 해결: 원형 Circular Progress 추가된 최종 버전

아래는 **기존 구조를 유지하면서** 원형 그라데이션 프로그레스를 정상적으로 표시하도록 수정한 버전입니다.

### 🔧 핵심 변경점

1. `CustomPaint` 를 Stack의 가장 아래에 추가
2. `animatedProgress` 를 Painter에 전달
3. strokeWidth는 기본값 유지 또는 필요 시 옵션화 가능
4. 이미지가 원형 프로그레스 중앙에 위치하도록 구조 정돈

---

## 📌 완성된 build() 버전

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


class GradientCircularProgressPainter extends CustomPainter {
  final double progress; // 0.0 ~ 1.0
  final Color startColor;
  final Color endColor;
  final double strokeWidth;
  final Color? backgroundColor; // 배경 원 색상 (선택)

  GradientCircularProgressPainter({
    required this.progress,
    required this.startColor,
    required this.endColor,
    this.strokeWidth = 10.0,
```

```
    this.backgroundColor,
  });

  @override
  void paint(Canvas canvas, Size size) {
    final Offset center = Offset(size.width / 2, size.height / 2);
    final double radius = math.min(size.width / 2, size.height / 2) - strokeWidth / 2;
    final Rect rect = Rect.fromCircle(center: center, radius: radius);

    if (backgroundColor != null) {
      final backgroundPaint = Paint()
        ..color = backgroundColor!
        ..style = PaintingStyle.stroke
        ..strokeWidth = strokeWidth;
      canvas.drawCircle(center, radius, backgroundPaint);
    }

    final progressPaint = Paint()
      ..shader = SweepGradient(
        colors: [startColor, endColor],
        startAngle: -math.pi / 2,
        stops: [0.0, 1.0],
        tileMode: TileMode.clamp,
      ).createShader(rect)
      ..style = PaintingStyle.stroke
      ..strokeWidth = strokeWidth
      ..strokeCap = StrokeCap.round;

    if (progress > 0.0) {
      canvas.drawArc(
        rect,
        -math.pi / 2,
        math.pi * 2 * progress.clamp(0.0, 1.0),
        false,
        progressPaint,
      );
    }
  }

  @override
  bool shouldRepaint(covariant GradientCircularProgressPainter oldDelegate) {
    return progress != oldDelegate.progress ||
        startColor != oldDelegate.startColor ||
        endColor != oldDelegate.endColor ||
        strokeWidth != oldDelegate.strokeWidth ||
        backgroundColor != oldDelegate.backgroundColor;
  }
}

class CircularProgressWithImage extends StatefulWidget {
  final double size;
  final double progress; // 0.0 ~ 1.0
  final Duration progressAnimationDuration;
```

```dart
  final String imagePath;

  const CircularProgressWithImage({
    super.key,
    required this.size,
    required this.progress,
    required this.imagePath,
    this.progressAnimationDuration = const Duration(milliseconds: 500),
  });

  @override
  State<CircularProgressWithImage> createState() => _CircularProgressWithImageState();
}

class _CircularProgressWithImageState extends State<CircularProgressWithImage>
    with SingleTickerProviderStateMixin {
  late final AnimationController _rotationController;

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

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

  @override
  Widget build(BuildContext context) {
    final double imageActualSize = widget.size * 0.55;

    return SizedBox(
      width: widget.size,
      height: widget.size,
      child: TweenAnimationBuilder<double>(
        tween: Tween<double>(begin: 0.0, end: widget.progress.clamp(0.0, 1.0)),
        duration: widget.progressAnimationDuration,
        builder: (context, animatedProgress, _) {
          return Stack(
            alignment: Alignment.center,
            children: [
              CustomPaint(
                size: Size(widget.size, widget.size),
                painter: GradientCircularProgressPainter(
                  progress: animatedProgress,
                  startColor: Colors.blueAccent,
                  endColor: Colors.lightBlueAccent,
```

```
                  strokeWidth: 10,
                  backgroundColor: Colors.grey[300],
                ),
              ),
              RotationTransition(
                turns: _rotationController,
                child: Image.asset(
                  widget.imagePath,
                  width: imageActualSize,
                  height: imageActualSize,
                  fit: BoxFit.contain,
                  errorBuilder: (context, error, stackTrace) {
                    print("Error loading image: $error");
                    return Icon(Icons.broken_image,
                        size: imageActualSize, color: Colors.grey);
                  },
                ),
              ),
            ],
          );
        },
      ),
    );
  }
}

```

## 🎯 기대 동작

| 기능 | 동작 여부 |
| --- | --- |
| 이미지 회전 | ✔ 정상 (AnimationController.repeat) |
| 프로그레스 바 표시 | ✔ Stack 아래에 CustomPaint 추가 |
| progress 애니메이션 | ✔ TweenAnimationBuilder 연결 |
| 그라데이션 원형 표현 | ✔ GradientCircularProgressPainter 적용 |
| 중앙 이미지 크기 조절 | ✔ 이미지가 CircularProgress보다 작게 조정 |

## 🧪 화면 구성 미리 설명

```
+------------------------------+
|           Progress Bar       |  <-- GradientCircularProgressPainter
|        +------------------+  |
|        |   Rotating Img   |  |  <-- RotationTransition + Image.asset
|        +------------------+  |
+------------------------------+
```