btn_setting.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/common/app_styles.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';

import '../../../common/app_colors.dart';
import '../../../common/app_strings.dart';

class SettingButton extends StatefulWidget {
  final bool state;
  final Function onTap;
  const SettingButton({super.key, required this.state, required this.onTap, });

  @override
  State<SettingButton> createState() => _SettingButtonState();
}

class _SettingButtonState extends State<SettingButton> {
  bool state = false;

  @override
  void initState() {
    state = widget.state;
    super.initState();
  }
  @override
  void didUpdateWidget(covariant SettingButton oldWidget) {
    if(oldWidget.state != widget.state) {
      setState(() {
        state = widget.state;
      });
    }
    super.didUpdateWidget(oldWidget);
  }

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () {
        widget.onTap();
      },
      child: Container(
        decoration: BoxDecoration(
            color: AppColors.FF007C5E,
            borderRadius: BorderRadius.circular(5)),
        height: 203,
```

```
        child: Center(
            child: Image.asset(
              state ? "${AppStrings.assetPath}img_stop.png" :
"${AppStrings.assetPath}img_stop.png",
              width: 90, height: 90,)
        ),
      ),
    );
  }

}
```

SettingButton 위젯 구조 분석

1. 위젯의 역할

`SettingButton`은 설정 화면에서 사용되는 **상태 기반 토글 버튼** 형태의 UI 컴포넌트다.
외부에서 `state`(Boolean)과 `onTap` 콜백을 받아 렌더링한다.

현재 구현 특성:

- 내부 상태(`state`)는 외부 파라미터를 초기값으로 사용
- 외부 `state` 값이 변경되면 `didUpdateWidget`에서 동기화함
- `InkWell`로 탭 이벤트 처리
- 버튼 내 중앙에 이미지를 표시하는 단순 UI

---

2. 코드 구성 요소 분석

2.1 StatefulWidget 구조

```
class SettingButton extends StatefulWidget {
  final bool state;
  final Function onTap;
}
```

외부 상태와 내부 상태가 혼합되어 있어 구조적으로 명확하지 않음.
해당 위젯은 **실질적으로 외부 상태를 그대로 표현하는 Controlled Component**인데 내부에서도 상태를 복제해 관리하고 있다.

→ 단일 Direction: 외부에서 주입하고 내부는 표현만 하는 형태가 더 안정적이다.

2.2 상태 동기화

```
@override
void initState() {
  state = widget.state;
  super.initState();
}
void didUpdateWidget(covariant SettingButton oldWidget) {
  if (oldWidget.state != widget.state) {
    setState(() {
      state = widget.state;
    });
  }
}
```

외부 상태 변화를 감지해 내부 상태 갱신하는 구조지만,
이 작업 자체가 필요 없는 설계다. 위젯이 상태를 "표현"만 한다면 내부 변수는 불필요하다.

2.3 이미지 Toggle 코드

현재 코드:

```
state ? "img_stop.png" : "img_stop.png"
```

두 경로가 동일하다.
버튼 토글 UI가 구현되지 않은 채 복사된 것으로 보임.
실제 의도는 `img_start.png` / `img_stop.png` 형태일 가능성이 높다.

2.4 UI 구조

```
InkWell
└── Container
    └── Center
        └── Image.asset
```

구조 자체는 단순하지만 다음 문제점이 있다.

문제점

- 버튼 색이 고정(AppColors.FF007C5E) → Theme와 분리됨
- 이미지 크기 값 직접 하드코딩
- Container+InkWell보다 Material+InkWell 구조가 더 자연스럽다
- stateful component로 분리할 필요가 없음

---

3. 설계적 관점에서의 문제점 요약

1. **StatefulWidget일 필요 없음**
   외부에서 상태를 받아 단순히 렌더링만 하면 StatelessWidget으로 충분함.
2. **내부 상태 중복 관리**
   동일한 데이터를 두 곳에서 관리 → 불필요.
3. **Toggle 이미지 구현 미완성**
   state가 true/false여도 동일 이미지 렌더링.
4. **확장성 부족**
   padding, radius, image 크기 등 커스터마이징 옵션 없음.
5. **Press ripple 효과 없음**
   Material + InkWell 조합 사용해야 정상적인 Ripple 적용 가능.

✔ Step3. 리팩토링 완전체 코드

3.1 Stateless 기반 구조 + 완전한 단방향 데이터 흐름

3.2 이미지 토글 정상화

3.3 확장성 있는 API 설계

3.4 Ripple 정상 적용

3.5 const 정제

3.6 Theme 기반 확장 가능성 유지

```dart
import 'package:flutter/material.dart';
import 'package:buyoil/common/app_colors.dart';
import 'package:buyoil/common/app_strings.dart';
import 'package:buyoil/common/app_styles.dart';

class SettingButton extends StatelessWidget {
  final bool state;
  final VoidCallback onTap;
  final double height;
  final double imageSize;
  final Color backgroundColor;
  final BorderRadiusGeometry borderRadius;

  const SettingButton({
    super.key,
    required this.state,
    required this.onTap,
    this.height = 203,
    this.imageSize = 90,
    this.backgroundColor = AppColors.FF007C5E,
    this.borderRadius = const BorderRadius.all(Radius.circular(5)),
```

```
  });

  @override
  Widget build(BuildContext context) {
    return Material(
      color: Colors.transparent,
      child: InkWell(
        borderRadius: borderRadius,
        onTap: onTap,
        child: Container(
          height: height,
          decoration: BoxDecoration(
            color: backgroundColor,
            borderRadius: borderRadius,
          ),
          child: Center(
            child: Image.asset(
              state
                  ? "${AppStrings.assetPath}img_stop.png"
                  : "${AppStrings.assetPath}img_start.png",
              width: imageSize,
              height: imageSize,
            ),
          ),
        ),
      ),
    );
  }
}
```

---

4. 리팩토링 기능별 설명

4.1 Stateless로 전환

SettingButton은 외부에서 상태를 전달받는 구조가 더 적합하고,
위젯 내부에서 상태를 유지할 이유가 없음.

4.2 Material + InkWell

Press ripple 효과가 정상적으로 동작함.

4.3 이미지 토글 정상화

`start` / `stop` 구조에 맞게 분기.

4.4 API 확장성 강화

- height
- imageSize
- backgroundColor

- borderRadius
  외부에서 스타일 조정 가능.

4.5 const 최적화

빌드 성능 및 렌더링 개선.

---

5. 구조 분석 기준에서 이 위젯의 위치

```
Presentation Layer
└── View Widgets
    └── Setting
        └── SettingButton.dart (UI용 Stateless Component)
```

역할:
사용자 입력(Action)을 트리거하며, 상태 표현만 담당하는 Dumb Widget

비즈니스 로직을 포함하지 않는 구조가 이상적이다.