STATE MANAGEMENT DASAR (FLUTTER)

A. Pendahuluan

Dalam Flutter, **state** adalah data yang dapat berubah seiring waktu dan memengaruhi tampilan (UI).

Contoh:

- Nilai counter
- Status login
- Data dari API
- Input form

Flutter bersifat **reactive**, artinya:

Jika **state berubah → UI akan diperbarui**

B. Konsep State

1. Apa itu State?

State adalah kondisi/data dari sebuah widget pada waktu tertentu.

Contoh state:

- Angka: `int counter = 0`
- Boolean: `bool isLoading = true`
- String: `String username = ""`

---

2. Jenis State di Flutter

a. StatelessWidget

- Tidak memiliki state
- UI **tidak berubah**
- Contoh: teks statis, icon

```
class MyText extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Text("Hello Flutter");
  }
}
```

---

b. StatefulWidget

- Memiliki state
- UI **bisa berubah**
- Menggunakan `setState()`

```
class CounterPage extends StatefulWidget {
  @override
  _CounterPageState createState() => _CounterPageState();
}
```

C. setState()

1. Pengertian setState

`setState()` adalah method untuk:

- Memberi tahu Flutter bahwa **state berubah**
- Memerintahkan Flutter untuk **me-render ulang UI**

Tanpa setState, UI tidak akan berubah

---

2. Contoh Sederhana setState

Counter App

```
class CounterPage extends StatefulWidget {
  @override
  _CounterPageState createState() => _CounterPageState();
}

class _CounterPageState extends State<CounterPage> {
  int counter = 0;

  void increment() {
    setState(() {
      counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text("Counter App")),
      body: Center(
        child: Text(
          counter.toString(),
          style: TextStyle(fontSize: 40),
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: increment,
        child: Icon(Icons.add),
      ),
    );
  }
}
```

---

3. Cara Kerja setState

1. State berubah di dalam `setState()`
2. Flutter memanggil ulang `build()`
3. UI diperbarui sesuai state terbaru

---

4. Kesalahan Umum

Mengubah state tanpa `setState`

```
counter++; // UI tidak berubah
```

Benar:

```
setState(() {
  counter++;
});
```

D. Konsep State Management

1. Apa itu State Management?

State Management adalah **cara mengelola dan membagikan state** agar:

- Kode rapi
- Mudah dipelihara
- Tidak duplikasi data

---

2. Masalah jika tanpa State Management

- Terlalu banyak `setState`
- Data sulit dikontrol
- Widget saling bergantung
- Kode sulit dibaca

---

3. State Lokal vs Global

State Lokal

- Digunakan di 1 halaman
- Contoh: counter, checkbox

```
int counter = 0;
```

State Global

- Digunakan di banyak halaman
- Contoh: user login, tema, data API

E. Best Practice Pengelolaan Data

1. Jangan Simpan Logic di build()

Salah:

```
@override
Widget build(BuildContext context) {
  int total = harga * jumlah;
  return Text(total.toString());
}
```

Benar:

```
int total = 0;

void hitungTotal() {
  total = harga * jumlah;
}
```

---

2. Gunakan Fungsi untuk setState

Buruk:

```
onPressed: () {
  setState(() {
    counter++;
  });
}
```

Baik:

```
void increment() {
  setState(() {
    counter++;
```

```
  });
}
```

---

3. State Sekecil Mungkin

- Jangan taruh semua state di satu widget
- Pecah widget jika perlu

Prinsip: **Small widget, small state**

---

4. Hindari setState Berlebihan

Terlalu sering:

```
setState(() {});
```

Gunakan hanya saat data berubah

---

5. Gunakan StatefulWidget Hanya Jika Perlu

Jika UI tidak berubah → gunakan `StatelessWidget`

F. Studi Kasus: Toggle Text

```
class ToggleText extends StatefulWidget {
  @override
  _ToggleTextState createState() => _ToggleTextState();
}

class _ToggleTextState extends State<ToggleText> {
  bool isVisible = true;

  void toggle() {
    setState(() {
      isVisible = !isVisible;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            if (isVisible)
              Text("Halo Flutter", style: TextStyle(fontSize:
24)),
            ElevatedButton(
              onPressed: toggle,
              child: Text("Toggle"),
            )
          ],
        ),
      ),
    );
  }
```

```
}
```

G. Kapan setState Tidak Cukup?

Gunakan state management lanjutan jika:

- Data digunakan banyak halaman
- Aplikasi besar
- Banyak logic bisnis

Contoh:

- Provider
- Riverpod
- Bloc
- GetX

Untuk pemula → setState sudah cukup

H. Ringkasan

- **State** = data yang berubah
- **setState()** = memberitahu Flutter untuk update UI
- Gunakan **StatefulWidget** dengan bijak
- Terapkan **best practice** agar kode bersih dan terstruktur

---

I. Latihan

1. Buat aplikasi counter dengan tombol + dan -
2. Buat form input nama, tampilkan hasilnya
3. Buat toggle tema terang/gelap sederhana

Berikut **contoh lengkap aplikasi Counter dengan tombol + dan − menggunakan Flutter (setState)**.

Aplikasi Counter (+ dan −)

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Counter App',
      home: CounterPage(),
```

```
    );
  }
}

class CounterPage extends StatefulWidget {
  @override
  State<CounterPage> createState() => _CounterPageState();
}

class _CounterPageState extends State<CounterPage> {
  int counter = 0;

  void increment() {
    setState(() {
      counter++;
    });
  }

  void decrement() {
    setState(() {
      if (counter > 0) {
        counter--;
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Counter + / -"),
        centerTitle: true,
      ),
      body: Center(
        child: Text(
          counter.toString(),
          style: const TextStyle(
            fontSize: 50,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
      bottomNavigationBar: Padding(
        padding: const EdgeInsets.all(16),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            ElevatedButton(
              onPressed: decrement,
              child: const Text(
                "-",
```

```
                style: TextStyle(fontSize: 30),
              ),
            ),
            ElevatedButton(
              onPressed: increment,
              child: const Text(
                "+",
                style: TextStyle(fontSize: 30),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Penjelasan Singkat

- `StatefulWidget` → karena nilai counter berubah
- `int counter` → state yang dikelola
- `setState()` → memberi tahu Flutter untuk update UI
- Tombol + menambah nilai
- Tombol − mengurangi nilai (dibatasi agar tidak minus)

Tugas Pengembangan (Opsional)

Coba kembangkan:

1. Tambahkan tombol **Reset**
2. Ubah warna angka saat counter > 10
3. Gunakan **FloatingActionButton** вместо tombol biasa