Mobile Application

Catatan Kuliah #13

Alauddin Maulana Hirzan, M. Kom
0607069401

Stateless dan **Stateful** Widget

Stateless dan **Stateful** Widget

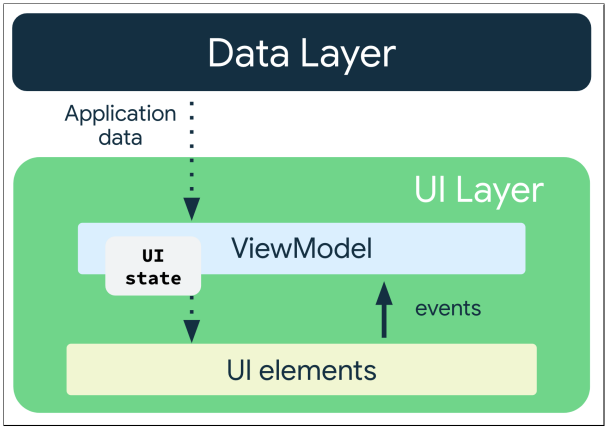
Pemrosesan Data Mobile

Dalam pemrograman menggunakan **Flutter** maka pastinya akan menggunakan variabel untuk melakukan input, modifikasi maupun output data. Konsep yang digunakan oleh Flutter adalah **Model**, **View**, dan **Controller** (**MVC**)

- **Model** : **Model** berkaitan dengan data
- **View** : **View** mengendalikan tampilan
- **Controller** : **Controller** mengendalikan aliran data dari **View**

Stateless dan **Stateful** Widget

Pemrosesan Data Mobile



Stateless dan **Stateful** Widget

Pemrosesan Data Mobile

Contoh:

- Dalam pemrograman menggunakan Kotlin, Objek **View** ditempeli dengan Listener yang bekerja sebagai **Controller**. Dengan keduanya data bisa diambil sebagai **Model**.

Keadaan saat ini yang di mana tampilan memiliki data yang bisa diambil, dimodifikasi, dan ditampilkan disebut sebagai **UI State**

Dalam pemrograman dengan Flutter, **widget** dapat beroperasi sebagai **Stateless** maupun **Stateful** Widget

Stateless dan **Stateful** Widget

Stateless Widget

Stateless Widget → Widget ini seperti namanya, tidak dapat diubah dan tidak mempertahankan status internal apa pun. Artinya, setelah dibuat, propertinya tidak dapat diubah. Widget ini murni didasarkan pada parameter input yang diberikan kepadanya dan tidak memiliki memori status sebelumnya.

Widget ini berguna untuk merepresentasikan komponen UI statis yang tidak memerlukan manajemen status internal apa pun. Widget ini ringan dan cepat karena tidak perlu menangani perubahan state atau membangun ulang dirinya sendiri.

Stateless dan **Stateful** Widget

Stateless Widget

```
class MyButton extends StatelessWidget {
  final String text;
  MyButton(this.text);

    @override
    Widget build(BuildContext context) {
    return RaisedButton(
      child: Text(text),
      onPressed: () {
        // Handle button press          },
    );   }}
```

Contoh kode ini akan menampilkan sebuah Tombol dengan tulisan dari variabel **text**. Namun karena bersifat **Stateless**, data yang ditampilkan hanya bisa satu kali saja

Stateless dan **Stateful** Widget

Stateless Widget

Ciri-ciri daripada Widget yang berjenis **Stateless** adalah

- *Immutable*
- *No Mutable State*
- Rebuild
- Cepat dan Ringan
- Murni Berdasarkan Masukan
- Independen

Stateless dan **Stateful** Widget

Stateless Widget

Immutable

Setelah dibuat, widget tanpa nama tidak dapat mengubah properti atau status internalnya. Properti widget tanpa nama ditetapkan selama penginstalan dan tetap konstan selama siklus hidup widget.

No Mutable State

Widget ini tidak mempertahankan status yang dapat diubah secara internal. Mereka hanya bergantung pada parameter input yang diberikan kepada mereka selama penginstalan.

Stateless dan **Stateful** Widget

Stateless Widget

Rebuild

Kapan pun widget induk dibangun ulang atau saat setState() dipanggil di widget induk, seluruh widget tanpa status akan dibangun ulang dari awal. Ini berarti metode build() dari widget tanpa nama dipanggil setiap kali widget perlu dirender.

Cepat dan Ringan

Widget tanpa kewarganegaraan ringan karena tidak memerlukan sumber daya tambahan untuk mengelola state. Widget ini lebih cepat dibandingkan dengan widget stateful karena tidak menangani perubahan state atau memicu pembangunan ulang.

Stateless dan **Stateful** Widget

Stateless Widget

Murni Berdasarkan Masukan

Widget tanpa status murni didasarkan pada parameter input yang diberikan kepadanya. Widget ini tidak memiliki memori apa pun tentang state atau interaksi sebelumnya. UI widget hanya ditentukan oleh properti input pada saat pembuatan.

Independen

Widget tanpa status tidak bergantung satu sama lain. Mereka tidak bergantung pada status widget lain atau berbagi data yang dapat diubah. Setiap widget tanpa status berdiri sendiri dan beroperasi berdasarkan propertinya sendiri.

Stateless dan **Stateful** Widget

Stateless Widget

Kapan **Stateless Widget** dapat digunakan?

- **Komponen UI bersifat statis**
  - Widget ini sangat cocok untuk membuat elemen UI statis seperti label, ikon, tombol, dan komponen visual lainnya yang tidak berubah berdasarkan interaksi pengguna atau pembaruan data.
- **Data Diteruskan sebagai Parameter Input**
  - Widget ini mengambil parameter masukan dalam konstruktornya dan menggunakannya untuk menentukan tampilan saat dibuat.

Stateless dan **Stateful** Widget

Stateless Widget

- **Tidak Perlu Penanganan Interaksi Pengguna**
  - Jika widget tidak memerlukan interaksi apa pun dari pengguna atau tidak merespons peristiwa pengguna. Widget tanpa nama tidak memiliki mekanisme pemanggilan balik internal atau penanganan peristiwa, karena mereka tidak
- **Performa yang lebih tinggi diinginkan**
  - Widget tanpa status ringan dan cepat karena tidak menangani perubahan status atau memicu pembangunan ulang. Jika kinerja menjadi perhatian atau jika Anda perlu merender sejumlah besar komponen UI statis, menggunakan widget tanpa nama dapat membantu mengoptimalkan proses rendering.

Stateless dan **Stateful** Widget

Stateful Widget

Di sisi lain, **Widget Stateful** memelihara dan mengelola status internalnya sendiri, yang dapat berubah seiring waktu. Widget ini digunakan ketika programmer perlu mengelola data dinamis dan merespons interaksi pengguna atau peristiwa lain.

Widget Stateful dapat diubah dan dapat memperbarui status internalnya dengan memanggil **setState()**, yang memicu pembuatan ulang widget.

Stateless dan **Stateful** Widget

Stateful Widget

Bagaimana cara kerja **Stateful Widget**?

- **Instansiasi Widget**
  - Ketika widget dengan stateful di-instansiasi, widget tersebut membuat instance dari kelas State yang terkait. Kelas **State** bertanggung jawab untuk mengelola status widget dan berisi variabel yang dapat berubah-ubah yang dapat berubah seiring waktu.
- **Metode build()**
  - Kelas State dari widget stateful mengesampingkan metode build(), yang menjelaskan UI widget berdasarkan statusnya saat ini. Metode build() dipanggil ketika widget perlu di-render atau dibangun ulang.

Stateless dan **Stateful** Widget

Stateful Widget

- **Manajemen Status**
  - Widget Stateful mengelola state mereka dengan menyimpan variabel yang dapat diubah dalam kelas State yang terkait. Variabel-variabel ini dapat diperbarui dan dimodifikasi sebagai respons terhadap interaksi pengguna, perubahan data, atau peristiwa lainnya.
- **Membangun ulang Widget**
  - Ketika setState() dipanggil, Flutter menandai widget sebagai kotor dan mengantri untuk dibangun ulang. Pada saat Flutter melakukan render pass, Flutter akan memanggil metode build() dari stateful widget.

Stateless dan **Stateful** Widget

Stateful Widget

- **Memperbarui UI**
  - Metode build() mengembalikan pohon widget yang merepresentasikan UI dari stateful widget. Flutter membandingkan pohon widget baru dengan pohon widget sebelumnya dan secara efisien hanya memperbarui bagian UI yang telah berubah.
- **Metode Siklus Hidup**
  - Widget Stateful juga menyediakan berbagai metode siklus hidup yang memungkinkan Anda melakukan tindakan pada titik tertentu dalam siklus hidup widget. Metode ini antara lain **initState()**, **dispose()**, dan **didUpdateWidget()**.

Stateless dan **Stateful** Widget

Stateful Widget

Ciri-ciri daripada **Stateful Widget** adalah

- **Mutable State**
  - Widget Stateful memelihara dan mengelola status internal mereka, yang dapat dimodifikasi dan diperbarui dari waktu ke waktu. Widget ini dapat menyimpan variabel yang dapat berubah-ubah yang dapat berubah sebagai respons terhadap interaksi pengguna, perubahan data, atau peristiwa lainnya.
- **Manajemen Status**
  - Widget Stateful memiliki metode untuk memperbarui state mereka. Metode yang paling umum digunakan adalah setState(). Ketika setState() dipanggil, Flutter mengetahui bahwa state telah berubah dan memicu pembangunan ulang widget, memperbarui UI berdasarkan state yang baru.

Stateless dan **Stateful** Widget

Stateful Widget

- **Membangun ulang sebagian**
  - Ketika widget yang telah dibangun ulang, Flutter secara efisien hanya memperbarui bagian UI yang telah berubah sejak pembangunan sebelumnya. Hal ini membantu mengoptimalkan kinerja dengan meminimalkan pembaruan UI yang tidak perlu.
- **Mempertahankan Status di Antara Build**
  - Tidak seperti widget tanpa status, widget dengan status mempertahankan statusnya di antara build. Artinya, saat widget dibangun ulang, status yang diperbarui akan dipertahankan, sehingga widget dapat merender UI berdasarkan status yang diperbarui.

Stateless dan **Stateful** Widget

Stateful Widget

- **Panggilan balik dan Penanganan Peristiwa**
  - Widget Stateful dapat menangani interaksi pengguna dan merespons peristiwa dengan mendefinisikan callback atau penangan peristiwa. Panggilan balik ini dapat dipicu oleh tindakan pengguna
- **Dinamis dan Interaktif**
  - Widget Stateful cocok untuk membangun komponen dengan perilaku dinamis dan fitur interaktif. Widget ini dapat merespons input pengguna, memperbarui statusnya, dan mencerminkan perubahan tersebut di UI.
- **Mempertahankan Cakupan Lokal**
  - Variabel-variabel tersebut tidak dapat diakses oleh widget lain kecuali secara eksplisit diteruskan

Stateless dan **Stateful** Widget

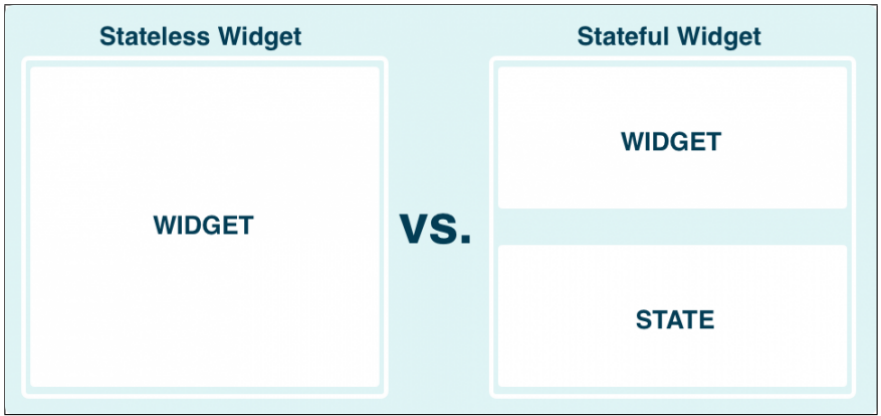
Stateful Widget

sample_page.dart

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

class SamplePage extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    return new SamplePageState();
  }

}

class SamplePageState extends State<SamplePage>{
  @override
  Widget build(BuildContext context) {
    // TODO: Implement build
  }

}
```

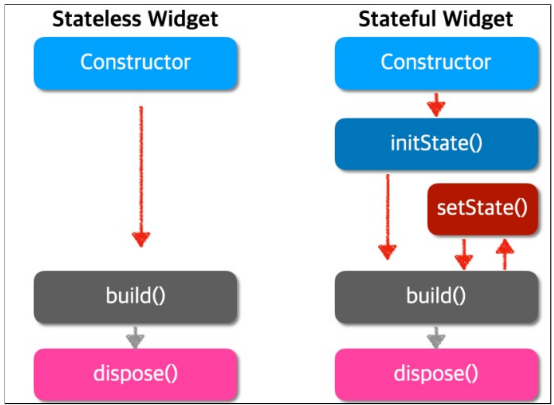
Stateless dan **Stateful** Widget

Perbandingan **Stateless** dan **Stateful** Widget



Stateless dan **Stateful** Widget

Perbandingan **Stateless** dan **Stateful** Widget



Stateless dan **Stateful** Widget

Perbandingan **Stateless** dan **Stateful** Widget

- **Stateful Widget**
  - Status awal widget diatur dalam metode createState() dari widget stateful, yang membuat sebuah instance dari kelas State yang terkait.
  - Ketika state perlu diperbarui, metode setState() dipanggil di dalam widget stateful untuk memperbarui
  - Flutter membandingkan pohon widget dan hanya memperbarui bagian UI yang telah berubah.
- **Stateless Widget**
  - Dibuat dan dirender sekali, dan UI-nya tetap konstan sepanjang siklus hidupnya.
  - Kapan pun widget induk dibangun ulang atau saat pembangunan ulang dipicu di widget induk, widget akan dibangun ulang seluruhnya.
  - Widget ini tidak memerlukan pembaruan atau pembangunan ulang lebih lanjut kecuali


THANK YOU