widget- state

Di bagian selanjutnya kita akan melihat bahwa kelas di Flutter menjadi widget ketika mewarisi dari StatelessWidget atau StatefulWidget.

StatelessWidget (kelas StatelessWidget) tidak dapat diubah. Setelah dibuat, itu tidak dapat diubah. Itu tidak mempertahankan statusnya. Widget tanpa kewarganegaraan dibuat

hanya sekali, dan tidak dapat dibangun kembali di lain waktu. 'build()' widget stateless metode dipanggil hanya sekali.

-> Widget tanpa kewarganegaraan. Gunakan widget semacam ini saat Anda perlu membuat UI yang

tidak akan berubah seiring waktu. Ini adalah blok "mandiri" yang tidak bergantung pada

peristiwa atau sumber eksternal; itu hanya bergantung pada konstruktor dan data internalnya.

StatefulWidget

Widget StatefulWidget (StatefulWidget) bisa berubah. Itu melacak negara.Ini membangun kembali beberapa kali selama masa pakainya. Metode `build()` widget Stateful adalah dipanggil berkali-kali.

->widget stateful. Gunakan widget semacam ini saat Anda perlu membuat UI yang

akan berubah seiring waktu. Dalam hal ini UI akan berubah secara dinamis karena

peristiwa eksternal seperti respons yang diterima dari permintaan HTTP atau panggilan balik yang dipicu

dengan menekan tombol.

STATE

Komunitas Flutter telah berjuang dengan masalah ini. Karena ini, mereka

telah menemukan beberapa solusi untuk menangani manajemen negara. Semua solusi ini memiliki satu aspek yang mereka bagikan: pemisahan model dan tampilan Kami akan mengimplementasikan aplikasi menggunakan 3 strategi pengelolaan negara yang berbeda:

1. Memperbarui UI menggunakan setState,

2. Melewati status di sekitar pohon widget, dengan bantuan widget Penyedia,

3. Alternatif untuk setState(), diimplementasikan dengan bantuan widget BlocBuilder.

**Interaksi routes dan navigasi**

Flutter memiliki mekanisme perutean imperatif, widget Navigator dan widget Router, yaitu mekanisme perutean deklaratif yang lebih idiomatis (mirip dengan metode build seperti yang digunakan dengan widget). Router adalah widget baru yang dapat Anda gunakan untuk membuka dan menutup halaman aplikasi. Navigator membungkus dan mengonfigurasi daftar halaman saat ini berdasarkan status aplikasi saat ini.

* **Dasar-dasar navigasi dan routes**

Mengatur informasi di beberapa layar adalah salah satu blok bangunan terpenting dari arsitektur apapun. Contoh yang sangat umum adalah di mana halaman pertama aplikasi Anda adalah login, jika pengguna memberikan kombinasi nama pengguna dan kata sandi yang benar, halaman selamat datang muncul. Cara paling umum untuk membuat halaman melibatkan penggunaan widget stateless atau stateful dengan Scaffold (CupertinoPageScaffold). Struktur folder untuk aplikasi :

- lib/

- route/

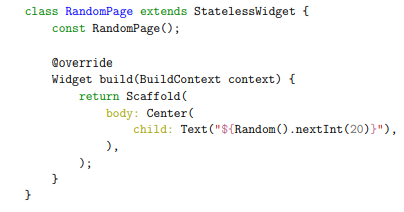
- widget/

- main.dart

- route.dart

* **Membuat routes**

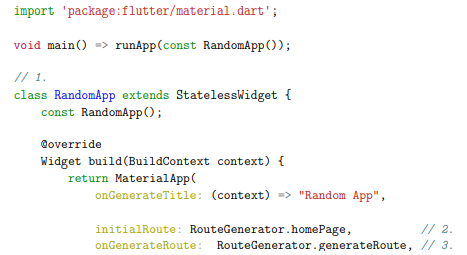
Untuk membuat aplikasi sederhana menggunakan dua layar : HomePage, route pertama yang muncul saat aplikasi dimulai dan RandomPage, route yang menampilkan nomor acak ditengah layar. Tata letak halaman akan dibuat dengan Scaffold yang nyaman dari perpustakaan bahan.

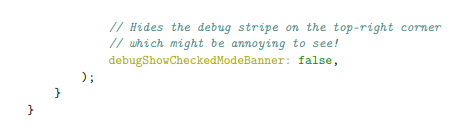
 

*Contoh kode program HomePage Contoh kode program HomePage*

* **File main.dart**

Struktur umum file main.dart mungkin terlihat seperti berikut ini. Penggunaan material alih-alih cupertino tidak relevan, itu hanya berarti kami menggunakan MaterialApp() sebagai ganti CupertinoApp().



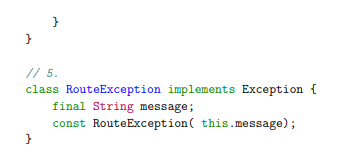


*Contoh kode program main.dart*

* **File routes.dart**

Memiliki satu kelas yang menangani seluruh perutean aplikasi Anda sangat bagus karena mencakup sepenuhnya prinsip tanggung jawab tunggal. Akibatnya, ketika harus berurusan dengan rute selalu membuka route.dart karena setiap logika terkait rute hanya ada di sana. Ini adalah kontrol "terpusat" dari rute yang menjaga kode tetap bersih. Contohnya adalah menggunakan bahan perpustakaan, jadi menggunakan pengembalian MaterialPageRoute().





*Contoh kode program routes.dart*

* **Pages diantara navigation**

Rekap singkat dari pekerjaan yang telah kami lakukan sejauh ini :

* pembuatan dua rute yang disebut DemoPage dan RandomPage;
* pengaturan rute di MaterialApp() dari main.dart;
* Pembuatan kelas RouteGenerator yang memetakan URI ke widget sehingga Kelas Navigator dapat membuka rute.
* **Navigator**

Kelas Navigator sudah ada sejak awal Flutter dan cara menggunakannya. Flutter akan memberikan pembaruan pada sistem perutean dengan:

* perubahan (beberapa di antaranya rusak);
* pop() tidak mengembalikan nilai lagi;
* properti isInitialRoute dari RouteSetting tidak digunakan lagi;
* adanya widget baru bernama Router.