MATERI

Hari-17-State Management

pada latihan kali ini kita akan berkenalan dengan state managemen di dart yaitu GetX

rangkuman materi ada di link url :

https://youtu.be/wuySrzOgevY

Tentang Get

- GetX adalah solusi ekstra-ringan dan powerful untuk Flutter. Ini mengkombinasikan state management dengan performa tinggi, injeksi dependensi yang cerdas, dan route management secara singkat dan praktis.
- GetX memiliki 3 prinsip dasar, yang menjadi prioritas untuk semua resource yang ada di dalamnya: **PRODUKTIFITAS, PERFORMA DAN ORGANISASI**
  - **PERFORMA:** GetX fokus pada performa dan konsumsi resource minimum. GetX tidak menggunakan Stream atau ChangeNotifier.
  - **PRODUKTIFITAS:** GetX menggunakan sintaks yang mudah dan nyaman. Tidak peduli apa yang akan anda lakukan, akan selalu ada cara yang lebih mudah dengan GetX. Ini akan menghemat waktu development, dan meng-ekstrak performa maksimum pada aplikasi anda. Umumnya, developer akan selalu berhubungan dengan penghapusan controller dari memori. Dengan GetX, ini tidak diperlukan, karena resource akan dihapus dari memori secara default ketika tidak digunakan. Jika anda ingin menyimpannnya kedalam memori, anda harus secara eksplisit mendeklarasikan "permanent: true" pada dependensi anda. Dengan begitu, selain menghemat waktu, anda juga mengurangi resiko memiliki dependensi yang tidak diperlukan dalam memori. Pemuatan dependensi juga bersifat "lazy" secara default.
  - **ORGANISASI:** GetX memungkinkan pemisahan View, Presentation Logic, Business Logic, Dependency Injection, dan Navigasi. Anda tidak perlu konteks untuk berpindah antar halaman. Jadi, anda tidak lagi bergantung pada widget tree (visualisasi) untuk hal ini. Anda tidak perlu konteks untuk mengakses controller/bloc melalui InheritedWidget. Dengan ini, anda benar benar memisahkan presentation logic dan business logic dari lapisan visual. Anda tidak perlu menginjeksi kelas Controller/Model/Bloc kedalam widget tree melalui multiprovider, untuk hal ini GetX menggunakan fitur dependency injection nya sendiri, memisahkan DI dari View secara total. Dengan GetX, anda tahu dimana harus mencari setiap fitur dalam aplikasi anda, memiliki kode yang bersih secara default. Ini selain untuk memfasilitasi maintenance, membuat pembagian modul, sesuatu yang hingga saat itu di Flutter tidak terpikirkan, sesuatu yang sangat mungkin. BLoC adalah permulaan awal dalam meng-organisir kode di Flutter, ini memisahkan business logic dari visualisasi. GetX adalah evolusi natural dari ini, tidak hanya memisahkan business logic, tapi juga presentation logic. Injeksi dependensi dan route juga dipisahkan sebagai bonus, dan lapisan data benar-benar terpisah secara menyeluruh. Anda tahu dimana semuanya berada, dan segalanya dengan cara yang lebih mudah daripada membuat sebuah hello world. GetX adalah cara termudah, praktis, dan scalable untuk membangun aplikasi dengan performa tinggi menggunakan Flutter SDK, dengan ekosistem besar di sekelilingnya yang bekerjasama secara sempurna, mudah dipahami untuk pemula, dan akurat untuk ahli. Aman, stabil, up-to-date, dan menawarkan banyak cakupan build-in API yang tidak tersedia di dalam default Flutter SDK.
- GetX tidak "bloated". Dirinya memiliki banyak fitur yang memungkinkan anda memulai programming tanpa mengkhawatirkan apapun, namun setiap fiturnya terletak didalam kontainer terpisah, dan hanya dimulai setelah digunakan. Jika anda hanya menggunakan State Management, hanya State Management yang akan di-compile. Jika anda hanya menggunakan routes, state management tidak akan di-compile.
- GetX memiliki ekosistem yang besar, komunitas yang juga besar, banyak kolaborator, dan akan di maintenance selama Flutter ada. GetX juga mampu berjalan dengan kode yang sama di Android, iOS, Web, Mac, Linux, Windows, dan server anda. **Juga memungkinkan untuk me-reuse kode yang dibuat di frontend ke backend dengan Get Server**.

Selain itu, seluruh proses development bisa di automasi secara menyeluruh, untuk keduanya (server dan frontend) menggunakan Get CLI.

Selain itu, untuk lebih meningkatkan produktifitas anda, kami memiliki ekstensi untuk VSCode dan ekstensi untuk Android Studio/Intellij

Instalasi

Tambahkan Get kedalam file `pubspec.yaml` anda:

```
dependencies:
  get:
```

Import get didalam file dimana get akan digunakan:

```
import 'package:get/get.dart';
```

Proyek "counter" yang dibuat secara default ketika membuat proyek Flutter memiliki lebih dari 100 baris (termasuk comment). Untuk menunjukkan kekuatan Get, kami akan mendemonstrasikan bagaimana cara membuat "counter" yang mengubah state setiap klik, berpindah, dan berbagi state antar halaman, semua dalam cara yang terorganisir, memisahkan business logic dari view, dalam HANYA 26 BARIS KODE TERMASUK COMMENT.

- Langkah 1: Tambahkan "Get" sebelum MaterialApp, mengubahnya menjadi GetMaterialApp

```
void main() => runApp(GetMaterialApp(home: Home()));
```

- Catatan: ini tidak mengubah MaterialApp bawaan Flutter, GetMaterialApp bukan sebuah MaterialApp yang dimodifikasi, itu hanyalah sebuah Widget yang telah dikonfigurasi sebelumnya, yang mana memiliki default MaterialApp sebagai child. Anda bisa mengkonfigurasinya secara manual, namun hal itu benar-benar tidak diperlukan. GetMaterialApp akan membuat route, menginjeksinya, menginjeksi translasi/terjemahan, dan semua yang anda butuhkan untuk navigasi route. Jika anda hanya menggunakan Get untuk manajemen state atau manajemen dependensi, tidak perlu menggunakan GetMaterialApp. GetMaterialApp diperlukan untuk route, snackbar, internasionalisasi/terjemahan, bottomSheet, dialog, dan high-level API yang berhubungan dengan route dan ketiadaan konteks.

- Catatan²: Langkah ini hanya diperlukan jika anda akan menggunakan manajemen route (`Get.to()`, `Get.back()` dan seterusnya). Jika anda tidak menggunakannya, langkah 1 tidak diperlukan.

- Langkah 2: Buat file baru untuk business logic dan taruh semua variabel, metode, dan kontroler didalamnya. Anda bisa membuat variabel apapun menjadi "observable" menggunakan notasi tambahan ".obs".

```
class Controller extends GetxController{
  var count = 0.obs;
  increment() => count++;
}
```

- Langkah 3: Buat file baru untuk View, gunakan StatelessWidget dan hemat penggunaan RAM, dengan Get, anda mungkin tidak perlu lagi menggunakan StatefulWidget.

```
class Home extends StatelessWidget {

  @override
  Widget build(context) {

    // Instansiasi kelas anda menggunakan Get.put() untuk membuatnya tersedia untuk seluruh "child" route dibawahnya.
    final Controller c = Get.put(Controller());

    return Scaffold(
      // Gunakan Obx(() => ...) untuk mengupdate Text() ketika `count` berubah.
      appBar: AppBar(title: Obx(() => Text("Clicks: ${c.count}"))),

      // Ganti 8 baris Navigator.push menggunan Get.to() agar lebih sederhana. Anda tidak perlu `context`.
      body: Center(child: ElevatedButton(
              child: Text("Go to Other"), onPressed: () => Get.to(Other()))),
      floatingActionButton:
          FloatingActionButton(child: Icon(Icons.add), onPressed: c.increment));
  }
}

class Other extends StatelessWidget {
  // Anda bisa meminta Get untuk menemukan kontroler yang digunakan di halaman lain dan redirect ke halaman itu.
  final Controller c = Get.find();

  @override
  Widget build(context){
     // Akses variabel `count` yang telah di update.
     return Scaffold(body: Center(child: Text("${c.count}")));
  }
}
```

Hasil:

Ini adalah proyek sederhana, namun sudah membuatnya terlihat jelas betapa powerful kemampuan yang dimiliki Get. Sepanjang proyek anda berkembang, perbedaan ini akan menjadi lebih signifikan.

Get di desain untuk bekerja dalam tim, namun juga memudahkan pekerjaan untuk developer perseorangan dan membuatnya menjadi lebih sederhana.

Tingkatkan deadline anda, antarkan semuanya tanpa kehilangan performa. Get bukan untuk semua orang, namun jika anda tersinggung dengan frasa tersebut, Get cocok untukmu!

State management

Get memiliki dua state manager berbeda: Simple state manager (kami menyebutnya GetBuilder) dan Reactive state manager (GetX/Obx)

Reactive State Manager

Reactive programming bisa meng-alienasi banya orang karena katanya, sulit dimengerti. GetX mengubah reactive programming menjadi sesuatu yang cukup sederhana:

- Anda tidak perlu membuat StreamController.
- Anda tidak perlu membuat StreamBuilder untuk setiap variabel.
- Anda tidak perlu membuat kelas untuk setiap state.
- Anda tidak perlu membuat get untuk sebuah value awal (initial value).
- Anda tidak perlu menggunakan generator kode.

Reactive programming dengan Get semudah menggunakan setState.

Bayangkan anda memiliki variabel nama, dan setiap kali anda mengubahnya, semua widget yang menggunakannya akan berubah secara otomatis.

Ini variabel count anda:

```
var name = 'Jonatas Borges';
```

Untuk membuatnya "observable", anda hanya perlu menambahkan ".obs" di belakangnya:

```
var name = 'Jonatas Borges'.obs;
```

Dan didalam UI, ketika anda ingin menampilkan value dan update tampilan ketika value itu berubah, cukup lakukan ini:

```
Obx(() => Text("${controller.name}"));
```

Selesai! *Sesederhana* itu.

Detail lebih lanjut mengenai state management

Baca penjelasan lebih lanjut tentang state management disini. Disana anda akan melihat contoh lebih banyak dan juga perbedaan diantara simple state manager dengan reactive state manager

Anda akan mendapatkan pemahaman yang baik tentang kekuatan dari GetX.

Route management

Jika anda ingin menggunakan routes/snackbars/dialogs/bottomsheets tanpa context, GetX luar biasa cocok untuk anda, lihat ini:

Tambahkan "Get" sebelum MaterialApp, mengubahnya menjadi GetMaterialApp

```
GetMaterialApp( // Sebelumnya: MaterialApp(
  home: MyHome(),
)
```

Pindah ke halaman baru:

```
Get.to(NextScreen());
```

Pindah ke halaman baru menggunakan nama. Baca detail lebih lanjut tentang penamaan route disini

```
Get.toNamed('/details');
```

Untuk menutup snackbar, dialog, bottomsheet, atau apapun yang normalnya anda tutup menggunakan Navigator.pop(context);

```
Get.back();
```

Untuk pergi ke halaman baru dan mencegah user kembali ke halaman sebelumnya (biasanya digunakan untuk SplashScreen, LoginScreen, dsb).

```
Get.off(NextScreen());
```

Untuk pergi ke halaman baru dan batalkan navigasi sebelumnya (berguna untuk shopping cart, polls, dan test).

```
Get.offAll(NextScreen());
```

Sadarkah bahwa anda tidak menggunakan context sama sekali untuk hal tersebut? Itu adalah keuntungan terbesar dalam menggunakan Get route management. Dengan ini, anda bisa mengeksekusi semua metode dari controller, tanpa ragu.

Detail lebih lanjut mengenai route management

Get bekerja dengan named route dan juga menawarkan kontrol dengan level yang lebih rendah untuk navigasimu! Dokumentasinya ada disini

Controller hanya dengan 1 baris kode, tanpa Provider context, tanpa inheritedWidget:

```
Controller controller = Get.put(Controller());
```

- Catatan: Jika anda menggunakan State Manager milik Get, harap untuk lebih memperhatikan Bindings api, yang mana akan membuat pengkoneksian View terhadap Controller jadi lebih mudah.

Daripada menginstansiasi kelas anda didalam kelas yang anda gunakan, cukup lakukan hal itu di dalam Get instance, ini akan membuatnya tersedia di semua tempat di Aplikasimu. Jadi anda bisa menggunakan controller (atau class Bloc) secara normal.

Tips: Dependency Management Get terpisah dari bagian lain dari package, jadi jika sebagai contoh aplikasi anda sudah menggunakan state manager (tidak peduli apapun itu), anda tidak perlu menulis ulang sama sekali, anda bisa menggunakan dependency injection tanpa masalah.

```
controller.fetchApi();
```

Bayangkan anda bernavigasi melewati route yang sangat banyak, dan anda membutuhkan data yang tertinggal didalam controller jauh di belakang route sebelumnya, anda akan butuh state manager dikombinasikan dengan Provider atau Get_it, benar kan? Tidak dengan Get. Anda hanya perlu meminta Get untuk "menemukan" controllernya, anda tidak perlu dependensi tambahan:

```
Controller controller = Get.find();
// Ya, terlihat seperti Sulap, Get akan menemukan controller anda, dan akan mengantarkannya ke lokasi anda.
// Anda bisa memiliki 1 juta controller terinisialisasi, Get akan selalu memberimu controller yang tepat.
```

Dan setelahnya anda bisa memperoleh data yang tertinggal sebelumnya:

```
Text(controller.textFromApi);


sumber: https://github.com/jonataslaw/getx/blob/master/README.id-ID.md#tentang-get
```

Rating - Feedback

Berikan Rating pada posting ini:

0

Berikan kritik dan saran..

Submit