

VERSION 1.0

JULI, 2023



PEMROGRAMAN MOBILE

STATE MANAGEMENT AND ARCHITECTURE FLUTTER – MODUL 2 MATERI

TIM PENYUSUN:

- DIDIH RIZKI CHANDRANEGERA, S.KOM., M.KOM.
- MUHAMMAD ZULFIQOR LILHAQ
- RIYAN PUTRA FIRJATULLAH

PRESENTED BY: LAB. INFORMATIKA UNIVERSITAS MUHAMMADIYAH MALANG

CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat memahami state management pada Framework Flutter.
2. Mahasiswa dapat memahami architecture pada Framework Flutter.

SUB CAPAIAN PEMBELAJARAN PRAKTIKUM

1. Mahasiswa dapat mengimplementasikan state management pada Framework Flutter.
2. Mahasiswa dapat mengimplementasikan architecture Framework Flutter.

KEBUTUHAN HARDWARE & SOFTWARE

1. PC/Laptop
2. IDE Android Studio/ Visual Studio Code
3. Flutter SDK: <https://docs.flutter.dev/release/archive?tab=windows>

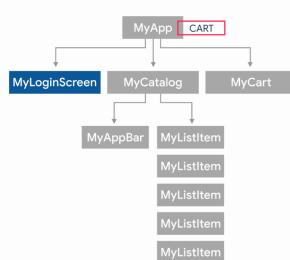
MATERI POKOK

State Management

State adalah informasi yang dapat dibaca saat widget dibuat, berubah atau termodifikasi selama penggunaan aplikasi. Jika ingin mengubah widget, state object harus diperbarui dengan menggunakan fungsi `setState()` yang tersedia pada widget `Stateful`. Fungsi `setState()` memungkinkan untuk menyetel properti objek status yang memicu penggambaran ulang UI [javatpoint-state management](#).



[ilustrasi state management](#)



[concept state management](#)

State manajemen adalah salah satu proses yang diperlukan dan paling sering terjadi dalam siklus hidup aplikasi. Flutter yang bersifat deklaratif membangun UI-nya berdasarkan state aplikasi yang terbaru [state management](#).

GetX

Adalah salah satu dari sekian banyak state management yang dimiliki flutter [list state management](#). GetX adalah state management ekstra-ringan dan powerful untuk Flutter yang mengkombinasikan performa tinggi, injeksi dependensi yang cerdas, dan route management secara singkat dan praktis [getx](#).

GetX principle:

1. Produktivitas

GetX menggunakan sintaks yang mudah dan nyaman sehingga banyak hal dapat dilakukan dengan lebih mudah. Ini akan menghemat waktu development, dan memberikan performa maksimum pada aplikasi.

Umumnya, developer akan selalu berhubungan dengan penghapusan controller dari memori. Dengan GetX resource akan dihapus dari memori secara default ketika tidak digunakan. Jika ingin menyimpannya ke dalam memori, cukup dengan mendeklarasikan secara eksplisit "permanent: true" pada sebuah dependensi. Sehingga dapat menghemat waktu dan mengurangi resiko penggunaan dependensi yang tidak diperlukan dalam memori. Pemuatan dependensi juga bersifat "lazy" secara default.

2. Performa

GetX fokus pada performa dan konsumsi resource minimum hal ini dikarenakan GetX tidak menggunakan Stream atau Change Notifier.

3. Organisasi

GetX memungkinkan pemisahan View, Presentation Logic, Business Logic, Dependency Injection, dan Navigasi. Developer tidak perlu lagi konteks untuk berpindah antar halaman sehingga dalam hal ini aplikasi tidak lagi bergantung pada widget tree (visualisasi).

Developer tidak perlu konteks untuk mengakses controller/bloc melalui InheritedWidget, sehingga presentation logic dan business logic benar-benar terpisahkan dari lapisan visual. Kelas Controller/Model/Bloc tidak perlu menginjeksi ke dalam widget tree melalui multiprovider, untuk hal ini GetX menggunakan fitur dependency injection nya sendiri yang terpisah dari View secara total.

Dengan GetX kode akan bersih secara default sehingga setiap fitur aplikasi dapat mudah dicari. Selain untuk memfasilitasi maintenance, hal ini dapat membuat pembagian modul pada Flutter menjadi sangat mungkin.

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 di dalam 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 front end ke backend dengan Get Server [getx](#).

Installation

Instalasi dilakukan cukup dengan menambahkan getx plugin pada file [pubspec.yaml](#) [getx](#).

1. Buka file [pubspec.yaml](#) pada project

-  nama_project
 -  pubspec.yaml

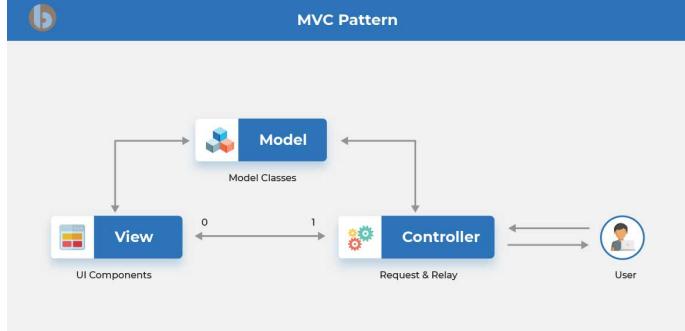
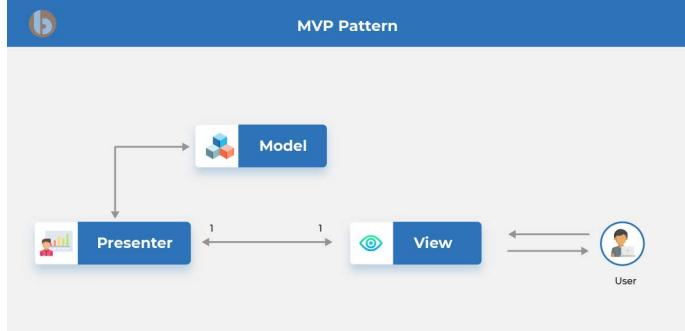
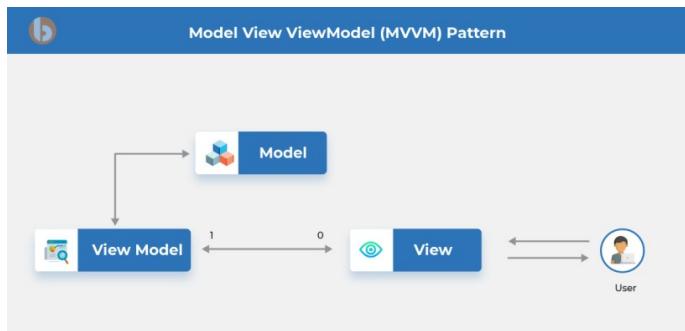
2. Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

dependencies: flutter: sdk: flutter <i># The following adds the Cupertino Icons font to your application. # Use with the CupertinoIcons class for iOS style icons.</i> cupertino_icons: ^1.0.2	dependencies: flutter: sdk: flutter <i># state management</i> get: 4.6.5 cupertino_icons: ^1.0.2
---	--

3. Selanjutnya **getx** sudah dapat dipakai oleh project

Architecture

Sebelum menggunakan getx, alangkah baiknya untuk memahami architecture terlebih dahulu. Memilih architecture project yang benar adalah salah satu kunci tepat untuk membangun aplikasi flutter. Architecture menentukan bagaimana kode aplikasi akan diatur dan bagaimana berbagai komponennya berinteraksi satu sama lain. Memilih architecture yang tepat dapat mempermudah pembuatan, pengujian, dan pemeliharaan aplikasi Anda dari waktu ke waktu [geeksforgeeks-architecture_flutter](#) [linkedin-architecture_flutter](#) [flutterawesome-architecture_flutter](#).

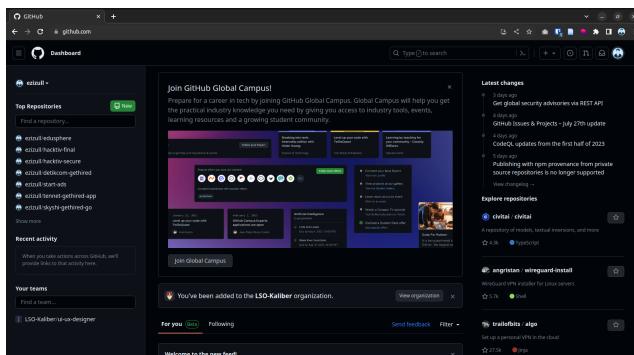
Architecture Pattern	
MVC Singkatan dari Model , View , Controller dimana user hanya akan berkomunikasi dengan Controller lalu baik Model , View , dan Controller dapat saling berkomunikasi sesuai permintaan user . contoh berdasar most start github: github-flutter_getx_mvc	 <p>MVC Pattern</p> <pre> graph TD Model[Model
Model Classes] --> View[View
UI Components] View -- 0 --> Controller[Controller
Request & Relay] Controller -- 1 --> Model Controller <--> User((User)) </pre>
MVP Singkatan dari Model , View , Presenter dimana user hanya akan berkomunikasi dengan View dan Presenter sebagai mediator dengan Model . contoh berdasar most start github: github-flutter_mvp	 <p>MVP Pattern</p> <pre> graph TD Model[Model
Model Classes] --> Presenter[Presenter
] Presenter -- 1 --> View[View
] View <--> User((User)) </pre>
MVVM Singkatan dari Model , View , View Model dimana user hanya akan berkomunikasi dengan View yang diteruskan ke Model dan View Model sesuai permintaan user . contoh berdasar most start github: github-flutter_getx_mvvm	 <p>Model View ViewModel (MVVM) Pattern</p> <pre> graph TD Model[Model
Model Classes] --> ViewModel[View Model
] ViewModel -- 1 --> View[View
] View <--> User((User)) </pre>

Template

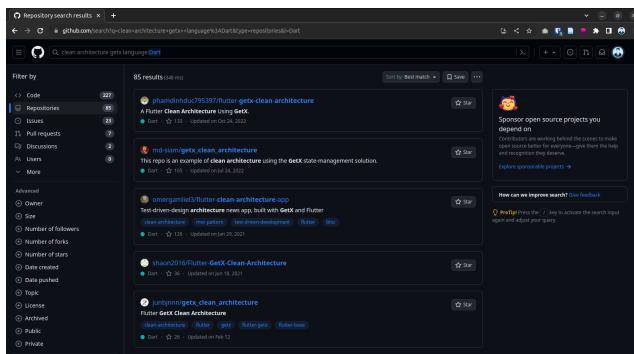
Saat ini sudah banyak architecture tercipta yang dapat diakses secara public. Terdapat beberapa cara untuk menggunakan sebuah architecture tutorial ini mencontohkan 2 diantaranya.

Pull Github

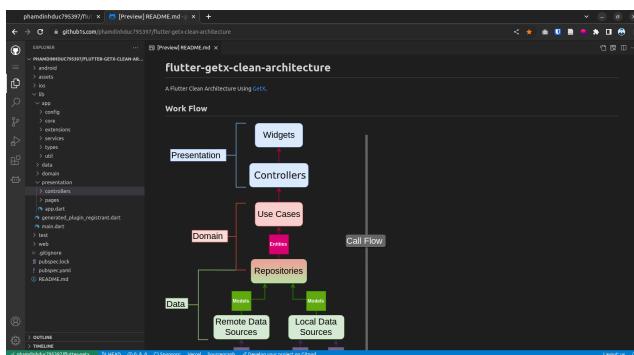
1. Buka github.com



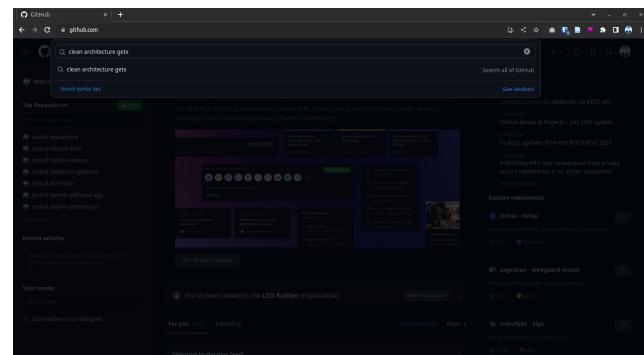
3. Pilih language **dart**



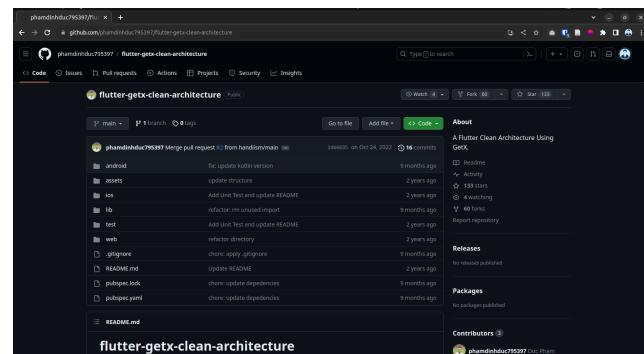
5. Untuk mengubah tampilan mirip vscode, cukup dengan mengubah github.com/bla-bla menjadi github1s.com/bla-bla



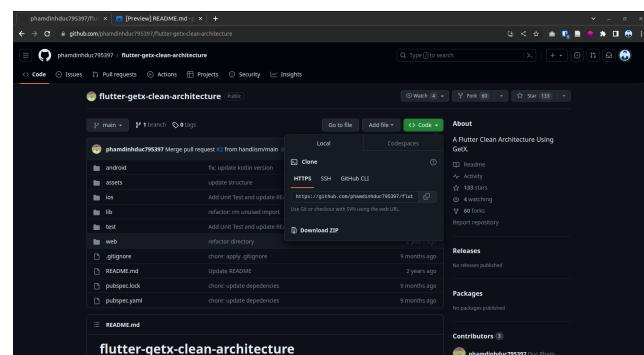
2. Search pattern yang diinginkan



4. Pilih salah satu pattern



6. Copy url repo github dari template yang diinginkan untuk kemudian di clone



7. Dengan terminal lakukan cloning di directory project yang diinginkan

```
git clone https://github.com/phamindduc795397/flutter-getx-clean-architecture.git
```

Get CLI

Adalah command line interface yang dapat dipakai untuk membuat seluruh infrastruktur getx dengan menggunakan template. get cli juga dapat sebagai solusi untuk mengurangi waktu setup controller, view, dan bindings [medium-get cli](#) [get cli](#).

Installation

- Pastikan **flutter** sudah terinstall pada sistem operasi

- Install get cli melalui terminal

```
flutter pub global activate get_cli
```

- Setelah itu **get cli** sudah dapat dipakai untuk berbagai project

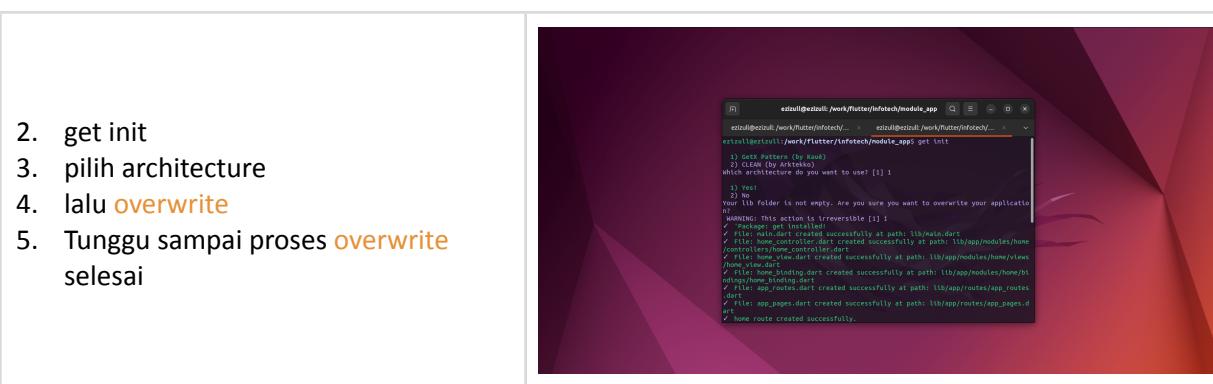
```
get --version
```

```
get help
```

Overwrite Project

Perlu diingat proses ini akan me **overwrite** folder **lib** atau mengganti seluruh yang ada pada **lib** sebelumnya

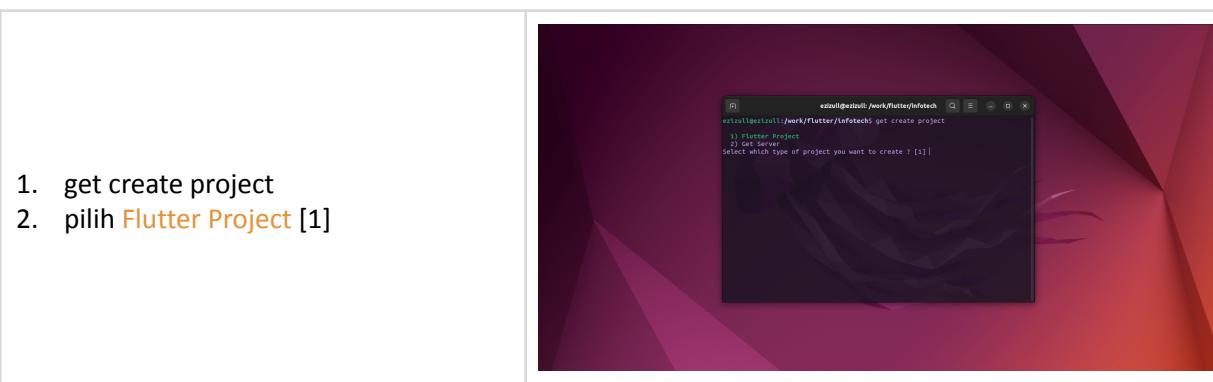
- Dengan command line pergi ke dalam folder project



- Open project yang telah di overwrite pada **IDE** atau **Text Editor**

Create Project

Get cli dapat dipakai untuk membuat project flutter dengan architecture tersedia beserta opsi lainnya



3. beri nama project
4. beri nama company
5. pilih bahasa untuk ios
6. pilih bahasa untuk android
7. pilih support null safe
8. pilih linter sebagai code analysis

```
ezizull@ezizull:~/work/flutter/infotech
$ flutter get create project
1) Flutter Project
2) Objetive-C
Select which type of project you want to create? [1] 1
what is your project name? getcli_project
what is your company domain? fluttercom.con_praktikun
1) Swift
which language do you want on ios? [1] 1
2) Objective-C
which language do you want to use on android? [1] 1
1) Java
2) Kotlin
what language do you want to use on android? [1] 1
1) Yes
2) No
do you want to use null safe? [1] 1
1) Yes
2) No
do you want to use some linter? [1] 1
```

9. Pilih **architecture** yang dipakai

terdapat 2 rekomendasi dari get cli yang dapat dipilih

- a. [getx pattern](#)
- b. [getx clean](#)

10. tutorial ini memilih **getx pattern** (optional)

```
ezizull@ezizull:~/work/flutter/infotech
$ flutter pub get
Running "flutter pub get" in getcli_project...
resolving dependencies...
async 2.5.0 (2.11.0 available)
charcode 1.1.3 (1.1.2 available)
collection 1.17.0 (1.18.0 available)
fs 0.10.5 (0.12.7 available)
http 0.13.4 (0.13.4 available)
matcher 0.12.13 (0.12.16 available)
meta 1.2.4 (1.9.3 available)
path 1.7.0 (1.7.1 available)
source_span 1.9.1 (1.10.0 available)
stack_trace 1.10.0 (1.11.1 available)
stream_channel 2.1.1 (2.1.2 available)
test 0.9.26 (0.9.27 available)
vm_service 4.6.16 (4.6.1 available)
✓ Package:flutter_lints installed
✓ file:analysis_options.yaml created successfully at path: analysis_options.yaml
[1] GETX pattern (by Kaud)
2) CLEAN (by Artekko)
which architecture do you want to use? [1] 1
1) Yes
2) No
Your lib folder is not empty. Are you sure you want to overwrite your application?
1) Yes
2) No
WARNING: This action is irreversible [1] 1
✓ file:main.dart created successfully at path: lib/main.dart
✓ file:lib/controller/home_controller.dart created successfully at path: lib/app/modules/home/controllers/home_controller.dart
✓ file:lib/view/home.dart created successfully at path: lib/app/modules/home/views/home.dart
✓ file:home_binding.dart created successfully at path: lib/app/modules/home/binding.dart
```

11. Pilih overwrite application

12. Tunggu sampai selesai

13. Open project yang telah dibuat pada IDE atau **Text Editor**

```
ezizull@ezizull:~/work/flutter/infotech
$ flutter pub get
Running "flutter pub get" in getcli_project...
resolving dependencies...
async 2.5.0 (2.11.0 available)
charcode 1.1.3 (1.1.2 available)
collection 1.17.0 (1.18.0 available)
fs 0.10.5 (0.12.7 available)
matcher 0.12.13 (0.12.16 available)
meta 1.2.4 (1.9.3 available)
path 1.7.0 (1.7.1 available)
source_span 1.9.1 (1.10.0 available)
stack_trace 1.10.0 (1.11.1 available)
stream_channel 2.1.1 (2.1.2 available)
test 0.9.26 (0.9.27 available)
vm_service 4.6.16 (4.6.1 available)
✓ Package:flutter_lints installed
✓ file:analysis_options.yaml created successfully at path: analysis_options.yaml
[1] GETX pattern (by Kaud)
2) CLEAN (by Artekko)
which architecture do you want to use? [1] 1
1) Yes
2) No
Your lib folder is not empty. Are you sure you want to overwrite your application?
1) Yes
2) No
WARNING: This action is irreversible [1] 1
✓ file:main.dart created successfully at path: lib/main.dart
✓ file:lib/controller/home_controller.dart created successfully at path: lib/app/modules/home/controllers/home_controller.dart
✓ file:lib/view/home.dart created successfully at path: lib/app/modules/home/views/home.dart
✓ file:home_binding.dart created successfully at path: lib/app/modules/home/binding.dart
```

Custom

Suatu Infrastructure juga dapat di custom sesuai kebutuhan masing-masing. Cukup dengan menambah, mengurangi atau memodifikasi struktural project.

Reactive State (RX)

Adalah sebuah tipe variabel **observable** atau reactive state sehingga variable memiliki kemampuan **stream**. Pada dasarnya **ragam** tipe data variabel ini sama dengan ragam variabel pada **umumnya** yang membedakan hanya pada penulisan dan **kemampuannya** saja. Terdapat tiga cara untuk inisialisasi **reactive state** [getx rx](#) [chorntorn-getx rx](#).

1. Menggunakan Rx{Type}

RxString	final name = RxString("");
RxBool	final isLoggedIn = RxBool(false);
RxInt	final count = RxInt(0);
RxDouble	final balance = RxDouble(0.0);
RxList<TypeData>	final items = RxList<String>([]);
RxMap<TypeData>	final myMap = RxMap<String, int>({});
Rx<Object>	—

2. Menggunakan Rx<Type>

RxString	final name = Rx<String>("");
RxBool	final isLoggedIn = Rx<Bool>(false);
RxInt	final count = Rx<Int>(0);
RxDouble	final number = Rx<double>(0);
RxList<TypeData>	final items = Rx<List<String>>([]);
RxMap<TypeData>	final myMap = Rx<Map<String, int>>({});
Rx<Object>	final user = Rx<User>();

3. Menggunakan .obs

RxString	final name = ".obs";
RxBool	final isLoggedIn = false.obs;
RxInt	final count = 0.obs;
RxDouble	final balance = 0.0.obs;
RxList<TypeData>	final items = <String>[].obs;
RxMap<TypeData>	final myMap = <String, int>{}.obs;
Rx<Object>	final user = User().obs;

GetMaterialApp

Adalah sebuah Widget yang telah dikonfigurasi sebelumnya, yang memiliki default MaterialApp sebagai child sehingga tidak diperlukan lagi konfigurasi secara manual. GetMaterialApp akan membuat route, menginjeksinya, menginjeksi translasi/terjemahan, dan semua yang dibutuhkan untuk navigasi route. GetMaterialApp diperlukan untuk route, snackbar, internasionalisasi/terjemahan, bottomSheet, dialog, dan high-level API yang berhubungan dengan route dan ketiadaan konteks [getx.getmaterialapp](#).

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

  @override
  Widget build(BuildContext context) {
    SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);

    return GetMaterialApp(
      debugShowCheckedModeBanner: false,
      initialRoute: "/",
      unknownRoute: GetPage(
        name: "/notfound",
        page: () => NotFoundPage(),
      ),
      getPages: [
        // root
        GetPage(
          name: "/",
          page: () => RootPage(),
        ),
        // other
        GetPage(
          name: "/second",
          page: () => SecondPage(),
        ),
        GetPage(
          name: "/third",
          page: () => ThirdPage(),
        ),
      ],
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      themeMode: ThemeMode.system,
      translationsKeys: const {
```

```
'en_US': {
  'welcome': 'Welcome',
  'page_root': 'Root Page',
  'page_login': 'Login Page',
  'page_register': 'Register Page',
  'page_notfound': 'Noutfound Page',
},
},
locale: const Locale('en', 'US'),
);
}
}
```

debugShowCheckedModeBanner	bool	Untuk menghilangkan banner pada aplikasi
initialRoute	String?	Screen pertama dari aplikasi
unknownRoute	GetPage?	Screen 404 atau page tidak tersedia
getPages	List<GetPage>	Kumpulan screen atau routes yang ada pada aplikasi
theme	ThemeData?	Untuk setup light theme mode
darkTheme	ThemeData?	Untuk setup dark theme mode
themeMode	ThemeData?	Setup yang dipakai aplikasi saat: light (terang), dark (gelap), system (mengikuti sistem)
translationsKeys	Map<String, Map<String, String>>?	Translation aplikasi atau sebagai kamus bahasa pada aplikasi
locale	Locale?	Lokasi bahasa yang dipakai

required

nullable

GetPage

Sebagian aplikasi memiliki beberapa screen dengan informasi yang berbeda-beda, di flutter screens atau pages disebut juga dengan routes [flutter routes](#). Getpage adalah class milik [getx](#) untuk menginisialisasi screens dan pages atau routes pada aplikasi [fluttercampus-navigate getx](#).

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

  @override
  Widget build(BuildContext context) {
    SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersiveSticky);

    return GetMaterialApp(
      /// other code ///

      getPages: [
        // root
        GetPage(
          name: "/",
          page: () => RootPage(),
        ),
        // other
        GetPage(
          name: "/second",
          page: () => SecondPage(),
        ),
        GetPage(
          name: "/third",
          page: () => ThirdPage(),
        ),
      ],
      /// other code ///
    );
  }
}
```

name	String	Nama dari route / page / screen / path url (untuk web)
------	--------	--

page	Widget Function()	Widget atau class widget yang akan ditampilkan
binding	Bindings?	Untuk mengkoneksikan View dan Controller secara mudah.
middlewares	List<GetMiddle ware>?	Untuk memantau event-event pada route lalu memberikan tindakan sesuai kebutuhan

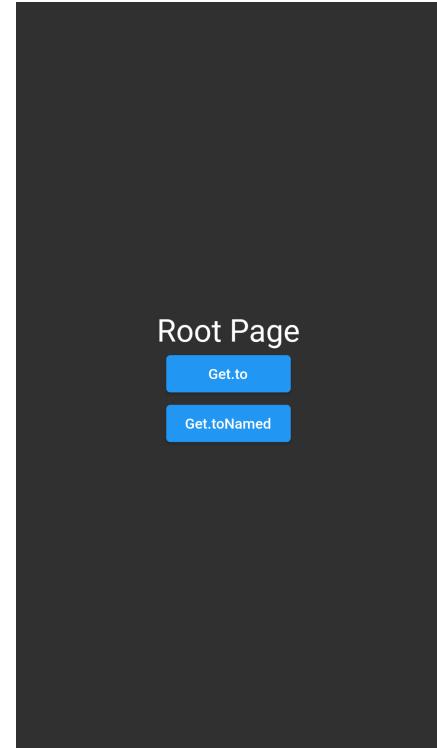
required

nullable

Setelah `GetMaterialApp` dan `GetPage` terkonfigurasi dengan benar fitur navigasi milik GetX sudah dapat dipakai [getx route](#).

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(SecondPage()),
                child: const Text("Get.to"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.toNamed('second'),
                child: const Text("Get.toNamed"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



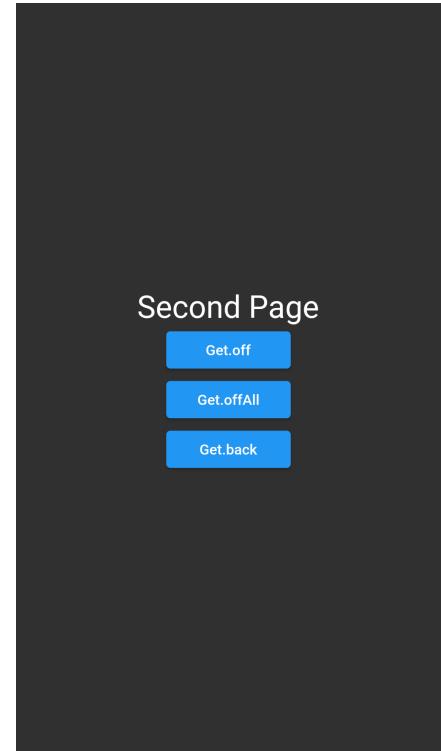
```

),
),
);
}
}

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

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text("Second Page", style: TextStyle(fontSize: 30)),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => Get.off(ThirdPage()),
 child: const Text("Get.off"),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => Get.offAll(ThirdPage()),
 child: const Text("Get.offAll"),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => Get.back(),
 child: const Text("Get.back"),
),
),
),
],
)
}
}

```



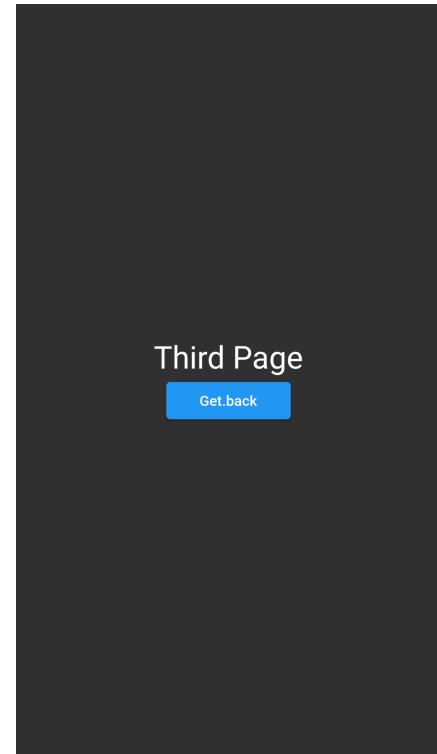
```

),
),
);
}
}

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

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text("Third Page", style: TextStyle(fontSize: 30)),
SizedBox(
width: 120,
child: ElevatedButton(
onPressed: () => Get.back(),
child: const Text("Get.back"),
),
),
],
),
),
),
);
}
}

```



Get.to(OtherScreen())	Berpindah halaman dengan memanggil widget class lain
Get.toNamed('other')	Berpindah halaman dengan sebuah nama route
Get.back()	Kembali kehalaman sebelumnya
Get.off(NextScreen())	Berpindah halaman dengan mengganti halaman sebelumnya (SplashScreen)
Get.offAll(NextScreen())	Berpindah halaman dan menghapus semua halaman sebelumnya (TestScreen)

GetController

Sebuah class pada getx yang berfungsi untuk mengontrol atau menampung informasi yang dipakai pada aplikasi [logrocket-getx](#).

```
class CountController extends GetxController {  
    final count = 0.obs;  
  
    Future<CountController> init() async {  
        const duration = Duration(seconds: 5);  
  
        Future.delayed(duration, () => ++count.value);  
        print("CountController:: Future init ${count.value}");  
  
        return this;  
    }  
  
    @override  
    void onInit() {  
        super.onInit();  
        ++count.value;  
        print("CountController:: onInit ${count.value}");  
    }  
  
    @override  
    void onReady() {  
        super.onReady();  
        ++count.value;  
        print("CountController:: onReady ${count.value}");  
    }  
  
    @override  
    void onClose() {  
        super.onClose();  
        --count.value;  
        print("CountController:: onClose ${count.value}");  
    }  
  
    void increment() {  
        count.value++;  
        print("CountController:: increment ${count.value}");  
    }  
}
```

```

class LazyCountController extends GetxController {
    final count = 0.obs;

    @override
    void onInit() {
        super.onInit();
        count.value++;
        print("LazyCountController:: onInit ${count.value}");
    }

    @override
    void onReady() {
        super.onReady();
        count.value++;
        print("LazyCountController:: onReady ${count.value}");
    }

    @override
    void onClose() {
        super.onClose();
        count.value--;
        print("LazyCountController:: onClose ${count.value}");
    }

    void increment() {
        count.value++;
        print("LazyCountController:: increment ${count.value}");
        update();
    }
}

```

obs	Observable merupakan Rx Types yang memiliki beragam metode dan operator internal
onInit	Function bawaan controller yang dipanggil segera setelah widget dialokasikan di memori
onReady	Function bawaan controller yang dipanggil setelah onInit(). bagus dipakai untuk navigation events, snackbar, dialogs, new route, atau async request

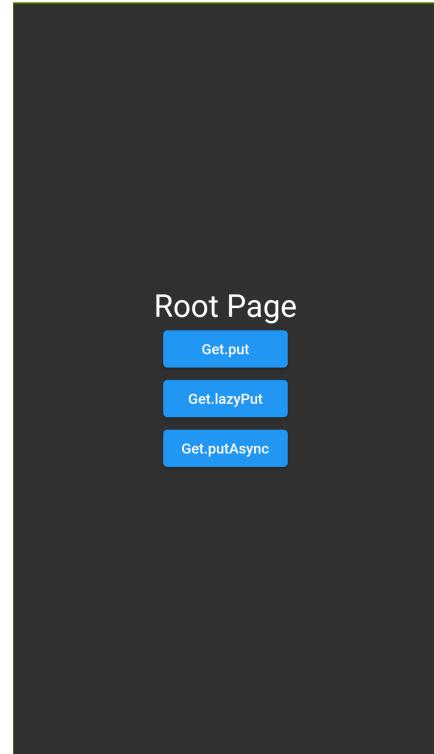
onClose	Function bawaan controller yang dipanggil sebelum onDelete() untuk melakukan dispose resources, seperti closing events, streams atau object yang berpotensi menyebabkan memory leaks
increment	Function pada umumnya hanya saja berada dalam controller

Penggunaan controller dapat dilakukan dengan beberapa cara tergantung kebutuhannya [learnflutter-get-controller](#).

Implementasi

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.min,
          children: [
            const Text("Root Page", style: TextStyle(fontSize: 30)),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(SecondPage()),
                child: const Text("Get.put"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(ThirdPage()),
                child: const Text("Get.lazyPut"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.to(FourthPage()),
                child: const Text("Get.putAsync"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



```

        ),
        ],
        ),
        ),
        );
    }
}

```

Get.put

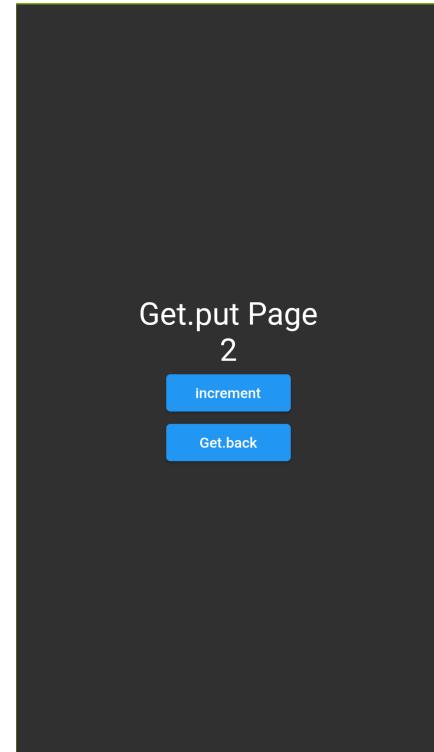
```

class SecondPage extends StatelessWidget {
SecondPage({super.key});

final controller = Get.put(CountController());

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text("Get.put Page", style: TextStyle(fontSize: 30)),
Obx(
() => Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => controller.increment(),
child: const Text("increment"),
),
),
SizedBox(
width: 120,
child: ElevatedButton(

```



```

        onPressed: () => Get.back(),
        child: const Text("Get.back"),
    ),
),
],
),
),
),
);
}
}

```

[GETX] Instance "CountController" has been created
I/flutter (4577): CountController::onInit 1
[GETX] Instance "CountController" has been initialized
[GETX] WARNING, consider using: "Get.to(() => Page())" instead of "Get.to(Page())".

Using a widget function instead of a widget fully guarantees that the widget **and** its controllers will be removed **from** memory when they are **no** longer used.

[GETX] GOING TO ROUTE /SecondPage
I/flutter (4577): CountController::onReady 2

Get.lazyPut

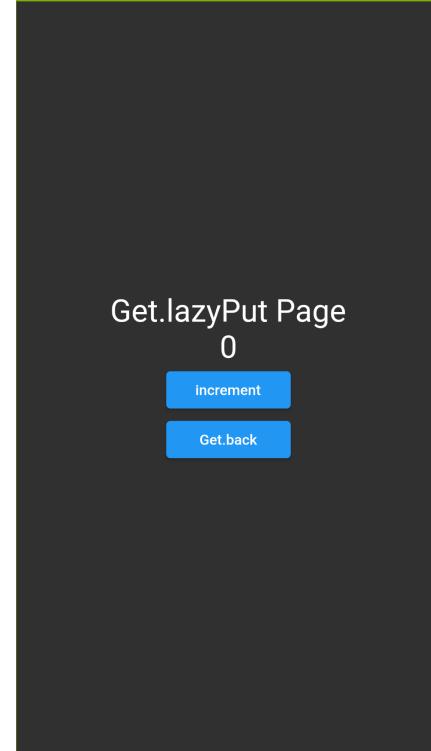
```

class ThirdPage extends StatelessWidget {
ThirdPage({super.key});

final leazyPut = Get.lazyPut(() => LazyCountController());
final controller = Get.find<LazyCountController>();

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: SizedBox(
            width: context.width,
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                mainAxisSize: MainAxisSize.min,
                children: [
                    const Text("Get.lazyPut Page", style: TextStyle(fontSize: 30)),
                    Obx(
                        () => Text(
                            "${controller.count.value}"),
                    )
                ],
            ),
        ),
    );
}

```



```
        style: const TextStyle(fontSize: 30),
    ),
),
SizedBox(
    width: 120,
    child: ElevatedButton(
        onPressed: () => controller.increment(),
        child: const Text("increment"),
    ),
),
SizedBox(
    width: 120,
    child: ElevatedButton(
        onPressed: () => Get.back(),
        child: const Text("Get.back"),
    ),
),
],
),
),
);
}
}
```

```
[GETX] GOING TO ROUTE /ThirdPage
[GETX] Instance "LazyCountController" has been created
I/flutter ( 4652): LazyCountController:: onInit 1
[GETX] Instance "LazyCountController" has been initialized
D/EGL_emulation( 4652): app_time_stats: avg=87106.72ms min=87106.72ms max=87106.72ms count=1
I/flutter ( 4652): LazyCountController:: onReady 2
D/EGL_emulation( 4652): app_time_stats: avg=407.39ms min=2.54ms max=3155.07ms count=8
[GETX] CLOSE TO ROUTE /ThirdPage
I/flutter ( 4652): LazyCountController:: onClose 1
[GETX] "LazyCountController" onDelete() called
[GETX] "LazyCountController" deleted from memory
```

Get.putAsync

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

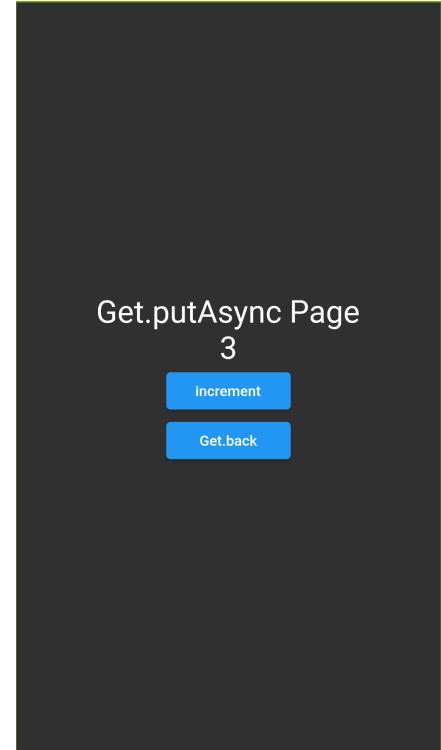
  await Get.putAsync(() => CountController().init());

  runApp(const Main());
}
```

```
class FourthPage extends StatelessWidget {
  FourthPage({super.key});

  final controller = Get.find<CountController>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            const Text("Get.putAsync Page", style: TextStyle(fontSize: 30)),
            Obx(
              () => Text(
                "${controller.count.obs}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => controller.increment(),
                child: const Text("increment"),
              ),
            ),
            SizedBox(
              width: 120,
```



```

        child: ElevatedButton(
            onPressed: () => Get.back(),
            child: const Text("Get.back"),
        ),
    ),
],
),
),
),
);
}
}
}

```

[GETX] WARNING, consider using: "Get.to(() => Page())" instead of "Get.to(Page())".

Using a widget function instead of a widget fully guarantees that the widget and its controllers will be removed from memory when they are no longer used.

[GETX] GOING TO ROUTE /FourthPage

D/EGL_emulation(4577): app_time_stats: avg=1357.76ms min=1357.76ms max=1357.76ms count=1

[GETX] Instance "CountController" has been created

I/flutter (4577): CountController:: onInit 1

I/flutter (4577): FourthPage:: Instance of 'CountController'

[GETX] Instance "CountController" has been initialized

I/flutter (4577): CountController:: onReady 2

I/flutter (4577): CountController:: Future init 3

Berikut kesimpulan sederhana dari beberapa cara penggunaan controller di atas berdasarkan output INTERFACE dan DEBUG CONSOLE.

Get.put(Controller())	Controller akan langsung menggunakan memory serta dapat disetup menjadi permanen di memory
Get.lazyPut(Controller())	Controller akan menggunakan memory saat dipakai dan menghapusnya dari memory saat tidak lagi dipakai
Get.putAsync(()=> Controller().init());	Untuk memanggil controller yang bersifat future serta dapat disetup menjadi permanen di memory

GetService

Adalah class mirip GetxController, yaitu memiliki `onInit()`, `onReady()`, `onClose()` akan tetapi tidak memiliki "logic" didalamnya. GetService hanya memberi tahu Sistem Dependency Injection GetX, bahwa subclass ini **TIDAK BISA** dihapus dari memori [getx.getservice](#).

1. Tambah package **GetStorage** pada `pubspec.yaml`

- `nama_project`
 - `lib`
 - `pubspec.yaml`

2. Ubah dari code **sebelum** (kiri) menjadi **sesudah** (kanan) lalu **save**.

<pre>dependencies: flutter: sdk: flutter # state management get: 4.6.5 cupertino_icons: ^1.0.2</pre>	<pre>dependencies: flutter: sdk: flutter # state management get: 4.6.5 # storage get_storage: 2.1.1 cupertino_icons: ^1.0.2</pre>
--	---

3. Package **GetStorage** sudah dapat digunakan pada project

Implementasi

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Get.putAsync(() => StorageService().init());

  runApp(const Main());
}
```

```
class StorageService extends GetxService {
  final storage = GetStorage("GetService");

  final TextEditingController inputControl = TextEditingController();

  RxString message = ".obs;

  Future<StorageService> init() async {
    await GetStorage.init("GetService");
    await writeStorage("GetService", "Welcome to GetService");

    message.value = await readStorage("GetService");

    storage.listenKey("GetService", (value) {
      message.value = value;
      print("StorageService:: ${message.value}");
    });
  }

  return this;
}

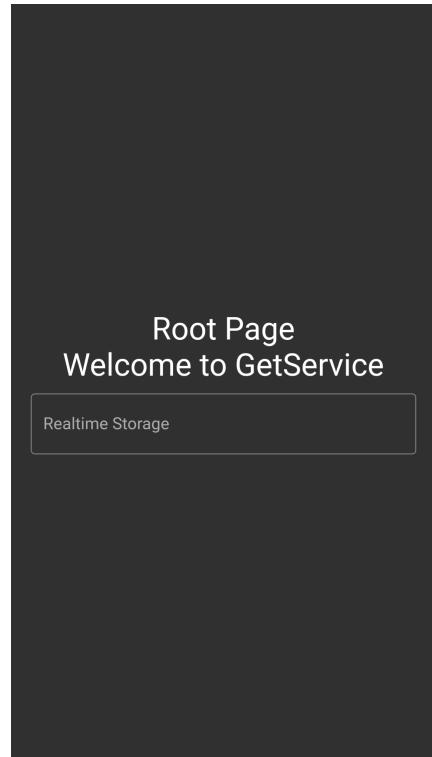
readStorage(String key) {
  return storage.read(key);
}

writeStorage(String key, dynamic value) {
  return storage.write(key, value);
}
```

```
class RootPage extends StatelessWidget {
RootPage({super.key});

final service = Get.find<StorageService>();

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
Text("Root Page", style: TextStyle(fontSize: 30)),
Obx(
() => Text(
"${service.message.obs}",
style: TextStyle(fontSize: 30),
),
),
),
Container(
margin: const EdgeInsets.only(top: 10),
padding: const EdgeInsets.symmetric(horizontal: 20),
child: TextField(
controller: service.inputControl,
onChanged: (value) {
service.writeStorage("GetService", value);
},
decoration: InputDecoration(
label: Text('Realtime Storage'),
border: OutlineInputBorder(
borderSide: BorderSide(
width: 1.5,
color: Theme.of(context).colorScheme.onBackground,
),
),
),
),
),
],
),
);
```



```

),
);
}
}

```

Get.find<StorageService>	S find<S>({String? tag})	Sebuah function dari Get yang dipakai untuk menggunakan GetService atau GetController yang sudah di inisialisasi
Obx	Widget	Sebuah class yang akan me return widget dengan metode Widget Function()
service.message.obs	RxString	Observable merupakan Rx Types yang memiliki beragam metode dan operator internal
storage.listenKey	void Function()	Untuk memantau perubahan pada storage yang memiliki key tertentu
message.value	void setValue(String val)	Value adalah sebuah function setter yang dari tipe data Rx

required

nullable

Bindings

Dapat digunakan untuk menginisialisasi pengontrol, repository, API, dan apapun yang diperlukan tanpa harus memanggilnya secara langsung pada page atau halaman [dev.to-getx create](#).

Implementasi

```
class CountBinding extends Bindings {
    @override
    void dependencies() {
        Get.put(CountController());
    }
}

class LazyCountBinding extends Bindings {
    @override
    void dependencies() {
        Get.lazyPut(() => LazyCountController());
    }
}
```

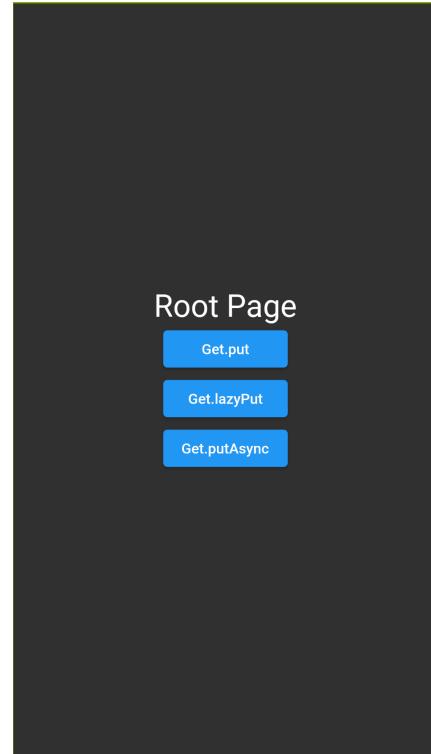
use

```
// root
GetPage(
    name: "/",
    page: () => RootPage(),
),

// other
GetPage(
    name: "/second",
    page: () => SecondPage(),
    binding: CountBinding(),
),
GetPage(
    name: "/third",
    page: () => ThirdPage(),
    binding: LazyCountBinding(),
),
GetPage(
    name: "/fourth",
    page: () => FourthPage(),
),
```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.max,
          children: [
            const Text("Root Page", style: TextStyle(fontSize: 30)),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.toNamed("/second"),
                child: const Text("Get.put"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.toNamed("/third"),
                child: const Text("Get.lazyPut"),
              ),
            ),
            SizedBox(
              width: 120,
              child: ElevatedButton(
                onPressed: () => Get.toNamed("/fourth"),
                child: const Text("Get.putAsync"),
              ),
            ),
          ],
        ),
      );
    }
}
```



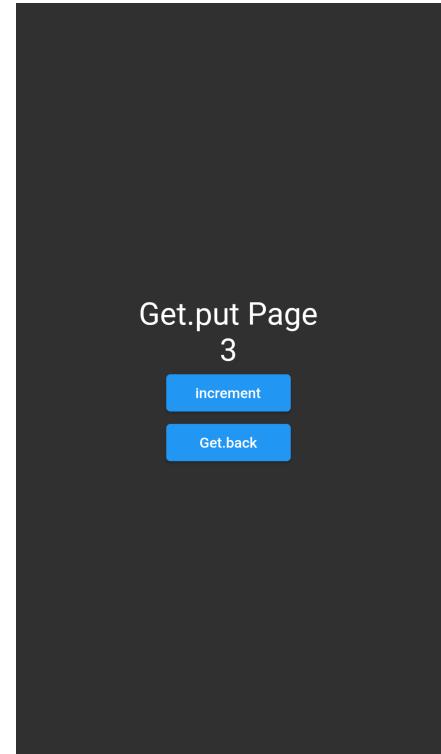
```

class SecondPage extends StatelessWidget {
SecondPage({super.key});

final controller = Get.find<CountController>();

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text("Get.put Page", style: TextStyle(fontSize: 30)),
Obx(
() => Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => controller.increment(),
 child: const Text("increment"),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => Get.back(),
 child: const Text("Get.back"),
),
),
],
),
);
}
}

```



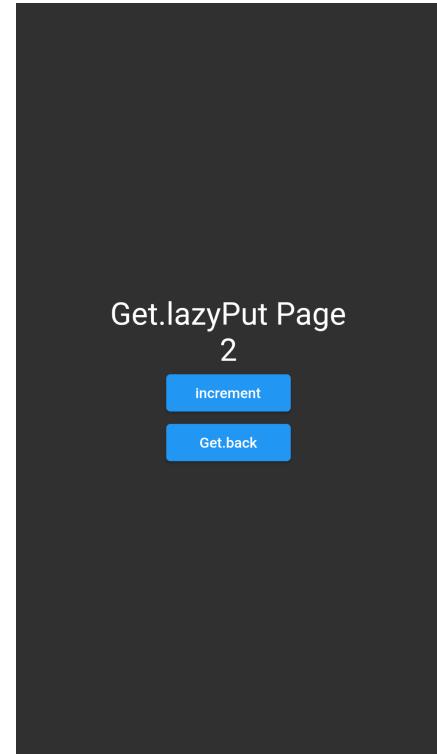
```

class ThirdPage extends StatelessWidget {
ThirdPage({super.key});

final controller = Get.find<LazyCountController>();

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
const Text("Get.lazyPut Page", style: TextStyle(fontSize: 30)),
Obx(
() => Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => controller.increment(),
child: const Text("increment"),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => Get.back(),
child: const Text("Get.back"),
),
),
],
),
);
}
}

```



GetMiddleware

Middleware memungkinkan untuk memantau peristiwa rute dan memicu tindakan yang sesuai. Ini memberikan cara mudah untuk menambahkan fungsionalitas atau melakukan tugas tertentu selama proses perutean [get middleware](#) [medium-get middleware](#).

Implementasi

```
class SessionMiddleware extends GetMiddleware {
  StorageService storageService = Get.find();

  @override
  RouteSettings? redirect(String? route) {
    storageService.message.refresh();
    if (!storageService.message.contains("user")) return const RouteSettings(name: "/");
    return null;
  }
}
```

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Get.putAsync(() => StorageService().init());

  runApp(const Main());
}
```

redirect

Dipanggil saat sedang menghubungkan ke sebuah route atau page. RouteSettings akan me redirect ke tujuan dan bila null maka tidak akan ada redirect.

use

```
GetPage(
  name: "/second",
  page: () => SecondPage(),
  binding: CountBinding(),
  middlewares: [SessionMiddleware()],
),
```

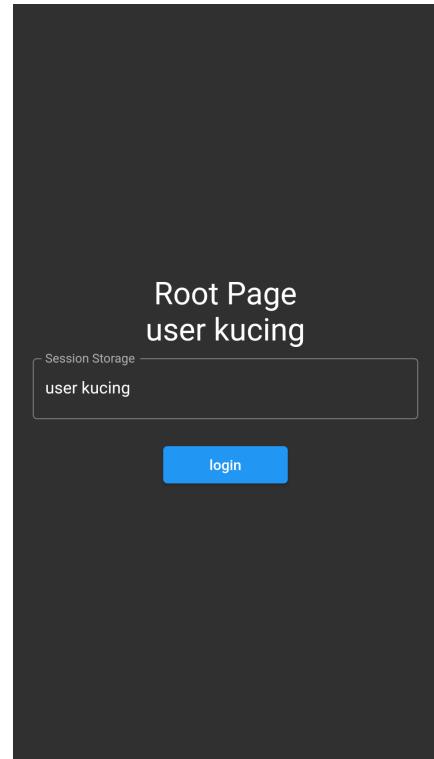
```

class RootPage extends StatelessWidget {
RootPage({super.key});

final service = Get.find<StorageService>();

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
Text("Root Page", style: TextStyle(fontSize: 30)),
Obx(
() => Text(
"${service.message.obs}",
style: TextStyle(fontSize: 30),
),
),
Container(
margin: const EdgeInsets.only(top: 10, left: 20, right: 20),
child: TextField(
controller: service.inputControl,
onChanged: (value) {
service.writeStorage("GetService", value);
},
decoration: InputDecoration(
label: Text('Session Storage'),
border: OutlineInputBorder(
borderSide: BorderSide(
width: 1.5,
color: Theme.of(context).colorScheme.onBackground,
),
),
),
),
),
Container(
width: 120,
margin: const EdgeInsets.only(top: 20),

```



```

        child: ElevatedButton(
            onPressed: () => Get.toNamed("/second"),
            child: const Text("login"),
        ),
    ),
],
),
),
),
);
}
}
}

```

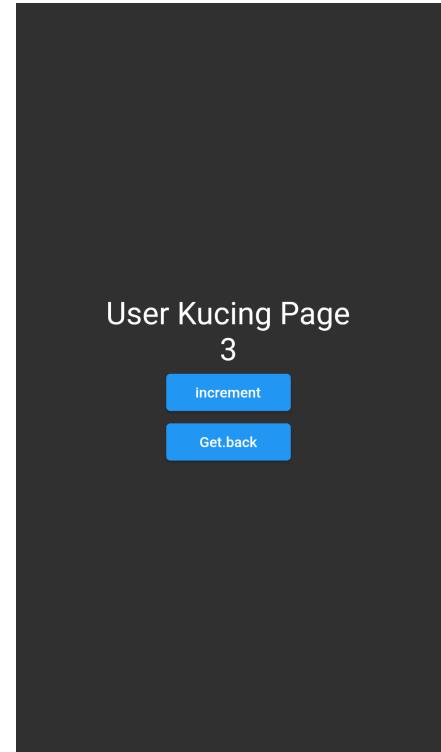
```

class SecondPage extends StatelessWidget {
SecondPage({super.key});

final controller = Get.find<CountController>();
final service = Get.find<StorageService>();

@Override
Widget build(BuildContext context) {
return Scaffold(
body: SizedBox(
width: context.width,
child: Column(
crossAxisAlignment: CrossAxisAlignment.center,
mainAxisAlignment: MainAxisAlignment.center,
children: [
Obx(
() => Text(
"${service.message.value.capitalize} Page",
style: const TextStyle(fontSize: 30),
),
),
Obx(
() => Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
),
SizedBox(
width: 120,
)
]
);
}
}
}

```



```
        child: ElevatedButton(
            onPressed: () => controller.increment(),
            child: const Text("increment"),
        ),
    ),
    SizedBox(
        width: 120,
        child: ElevatedButton(
            onPressed: () => Get.back(),
            child: const Text("Get.back"),
        ),
    ),
],
),
),
);
}
}
```

GetMiddleware dipakai untuk melakukan **authentication**, **authorization** atau validation role. Misal sudah login user dapat **menuju** ke suatu page atau user yang memiliki suatu role dapat mengakses suatu page sedangkan yang lain tidak.

GetView

Adalah sebuah const Stateless widget yang memiliki getter controller untuk controller yang telah diinisialisasi. Untuk menggunakan `GetView` ganti `StatelessWidget` atau `StatefulWidget` dengan `GetView<SpesifikController>` [get view](#).

Implementasi

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Get.putAsync(() => StorageService().init());

  runApp(const Main());
}
```

```
class ViewController extends GetxController {
  final service = Get.find<StorageService>();

  final count = 0.obs;

  void increment() {
    count.value++;
    print("ViewController:: increment ${count.value}");
    update();
  }
}
```

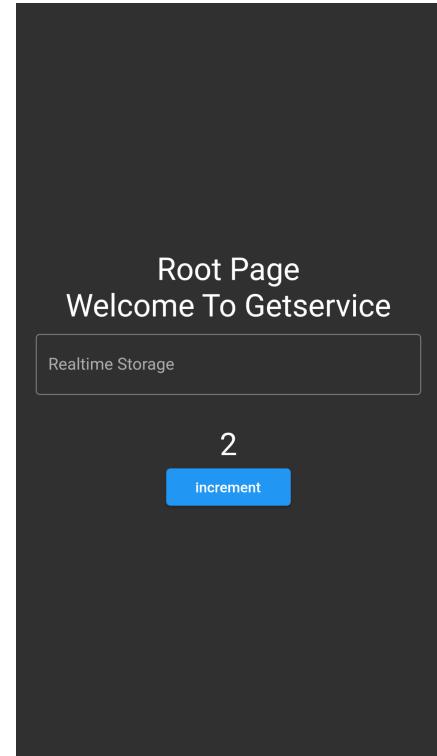
```
class ViewBinding extends Bindings {
  @override
  void dependencies() {
    Get.put(ViewController());
  }
}
```

```
GetPage(
  name: "/",
  page: () => const RootPage(),
  binding: ViewBinding(),
),
```

use

```
class RootPage extends GetView<ViewController> {
  const RootPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: context.width,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          mainAxisSize: MainAxisSize.min,
          children: [
            const Text(
              "Root Page",
              style: TextStyle(fontSize: 30),
            ),
            Obx(
              () => Text(
                "${controller.service.message.value.capitalize}",
                style: const TextStyle(fontSize: 30),
              ),
            ),
            Container(
              margin: const EdgeInsets.only(top: 10),
              padding: const EdgeInsets.symmetric(horizontal: 20),
              child: TextField(
                controller: controller.service.inputControl,
                onChanged: (value) {
                  controller.service.writeStorage("GetService", value);
                },
                decoration: InputDecoration(
                  label: const Text('Realtime Storage'),
                  border: OutlineInputBorder(
                    borderSide: BorderSide(
                      width: 1.5,
                      color: Theme.of(context).colorScheme.onBackground,
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```



```
),
Container(
margin: const EdgeInsets.only(top: 30),
child: Column(children: [
Obx(
() => Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => controller.increment(),
child: const Text("increment"),
),
),
]),
],
),
),
);
}
}
```

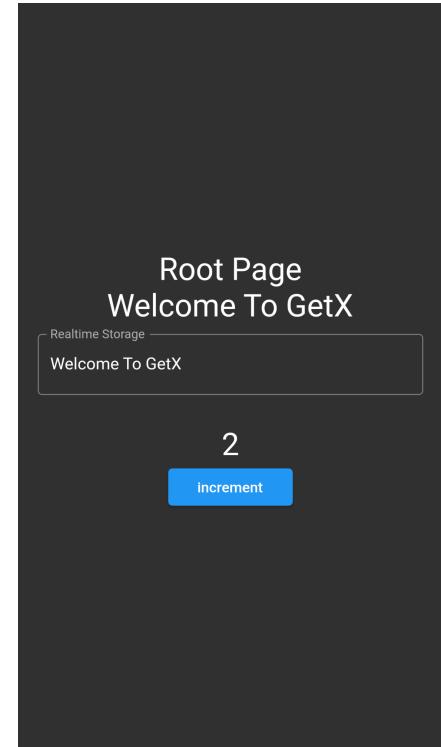
Manfaat dari GetView adalah penggunaan `final controller = Get.find<SpesifikController>();` dapat diganti cukup dengan langsung menggunakan `controller` saja. Tentu saja class controller yang dimaksud adalah apa yang ada pada `GetView<SpesifikController>`.

GetX

Bukan hanya nama package yang dipakai saat ini, **GetX** juga merupakan nama class di dalam package itu sendiri lebih tepatnya sebuah **builder** yang akan langsung mengupdate state bila terdapat perubahan. Class ini bukan hanya memantau variable melainkan **flow** sehingga **GetX** pasti kan mengupdate **UI** saat informasi pada specific object berubah [chornthorn-getx getx](#).

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

  @override
  Widget build(BuildContext context) {
    return GetX<ViewController>(
      builder: (controller) {
        return Scaffold(
          body: SizedBox(
            width: context.width,
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              mainAxisSize: MainAxisSize.min,
              children: [
                const Text(
                  "Root Page",
                  style: TextStyle(fontSize: 30),
                ),
                Text(
                  controller.service.message.value,
                  style: const TextStyle(fontSize: 30),
                ),
                Container(
                  margin: const EdgeInsets.only(top: 10),
                  padding: const EdgeInsets.symmetric(horizontal: 20),
                  child: TextField(
                    controller: controller.service.inputControl,
                    onChanged: (value) {
                      controller.service.writeStorage("GetService", value);
                    },
                    decoration: InputDecoration(
                      label: const Text('Realtime Storage'),
                      border: OutlineInputBorder(
                        borderSide: BorderSide(
                          width: 1.5,
                          color: Theme.of(context).colorScheme.onBackground,
                        ),
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ),
        );
      },
    );
  }
}
```



```
        ),  
        ),  
        ),  
        ),  
        ),  
        ),  
        Container(  
            margin: const EdgeInsets.only(top: 30),  
            child: Column(children: [  
                Text(  
                    "${controller.count.obs}",  
                    style: const TextStyle(fontSize: 30),  
                ),  
                SizedBox(  
                    width: 120,  
                    child: ElevatedButton(  
                        onPressed: () => controller.increment(),  
                        child: const Text("increment"),  
                    ),  
                ),  
            ]),  
        ),  
    ],  
),  
);  
},  
);  
}  
}
```

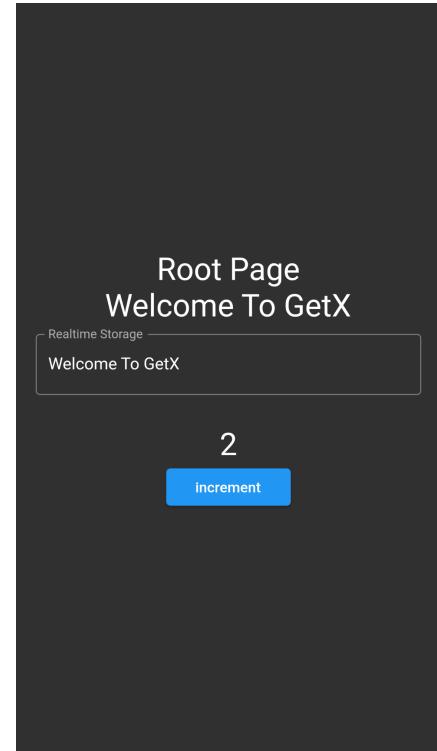
Untuk menggunakan GetX builder diperlukan spesifik `GetxServiceMixin` (`GetxController` atau `GetxService`) dengan menambahkan `<SpesifikGetxServiceMixin>` di samping kanan dari `GetX`. Untuk menggunakan `controller` dapat memanfaatkan nama variabel dari `builder`: `(controller)`. Tentu saja class controller yang dimaksud adalah apa yang ada pada `GetX<SpesifikGetxServiceMixin>` .

GetBuilder

Dipakai untuk mengontrol sebuah `group state` atau informasi dari controller. Semisal user menambah 30 produk ke keranjang, lalu user menghapus satu otomatis list produk akan terupdate, total biaya akan terupdate dan informasi lainnya juga ikut terupdate. GetBuilder akan mengupdate group state ini dalam satu waktu tanpa perlu "computational logic" dari hal itu [getx getbuilder](#).

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

  @override
  Widget build(BuildContext context) {
    return GetBuilder<ViewController>(builder: (controller) {
      return Scaffold(
        body: SizedBox(
          width: context.width,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            mainAxisSize: MainAxisSize.min,
            children: [
              const Text(
                "Root Page",
                style: TextStyle(fontSize: 30),
              ),
              Text(
                "${controller.service.message.value.capitalize}",
                style: const TextStyle(fontSize: 30),
              ),
              Container(
                margin: const EdgeInsets.only(top: 10),
                padding: const EdgeInsets.symmetric(horizontal: 20),
                child: TextField(
                  controller: controller.service.inputControl,
                  onChanged: (value) {
                    controller.service.writeStorage(".GetService", value);
                  },
                  decoration: InputDecoration(
                    label: const Text('Realtime Storage'),
                    border: OutlineInputBorder(
                      borderSide: BorderSide(
                        width: 1.5,
                        color: Theme.of(context).colorScheme.onBackground,
                      ),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      );
    });
  }
}
```



```
        ),
        ),
        ),
        ),
Container(
margin: const EdgeInsets.only(top: 30),
child: Column(children: [
Text(
"${controller.count.obs}",
style: const TextStyle(fontSize: 30),
),
SizedBox(
width: 120,
child: ElevatedButton(
 onPressed: () => controller.increment(),
child: const Text("Increment"),
),
),
]),
],
),
),
),
);
});
});
}
});
```

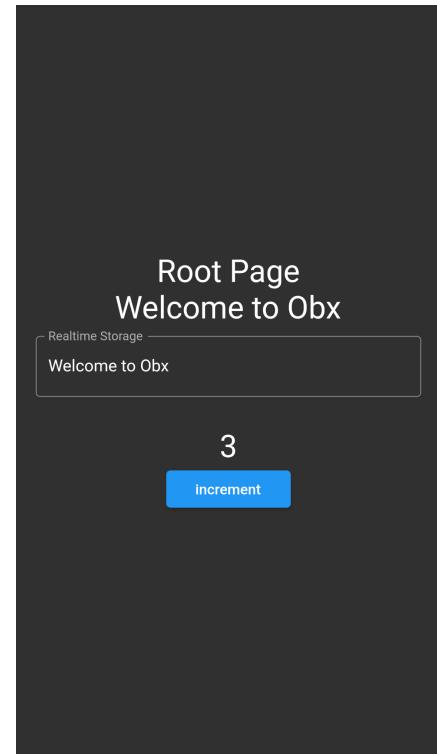
Untuk menggunakan GetBuilder diperlukan spesifik GetxController dengan menambahkan `<SpesifikController>` di samping kanan dari `GetBuilder`. Untuk menggunakan `controller` dapat memanfaatkan nama variabel dari `builder: (controller)`. Tentu saja class controller yang dimaksud adalah apa yang ada pada `GetBuilder<SpesifikController>`.

Obx | Obs

Obx adalah widget yang mengupdate UI saat terjadi perubahan state. Cara Obx sama seperti **statefull** yang membedakan kita tidak perlu lagi melakukan `setState(() {});` untuk mengupdate state. **Obs** adalah observable variable dari Reactive programming atau [RX {Type}](#).

```
class RootPage extends GetView<ViewController> {
  const RootPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Obx(
      () => Scaffold(
        body: SizedBox(
          width: context.width,
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            mainAxisSize: MainAxisSize.min,
            children: [
              const Text(
                "Root Page",
                style: TextStyle(fontSize: 30),
              ),
              Text(
                controller.service.message.value,
                style: const TextStyle(fontSize: 30),
              ),
              Container(
                margin: const EdgeInsets.only(top: 10),
                padding: const EdgeInsets.symmetric(horizontal: 20),
                child: TextField(
                  controller: controller.service.inputControl,
                  onChanged: (value) {
                    controller.service.writeStorage(".GetService", value);
                  },
                  decoration: InputDecoration(
                    label: const Text('Realtime Storage'),
                    border: OutlineInputBorder(
                      borderSide: BorderSide(
                        width: 1.5,
                        color: Theme.of(context).colorScheme.onBackground,
                      ),
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```



```
        ),
        ),
        ),
      Container(
        margin: const EdgeInsets.only(top: 30),
        child: Column(children: [
          Text(
            "${controller.count.obs}",
            style: const TextStyle(fontSize: 30),
          ),
          SizedBox(
            width: 120,
            child: ElevatedButton(
              onPressed: () => controller.increment(),
              child: const Text("increment"),
            ),
          ),
        ]),
      ),
    ],
  ),
);
}
}
```

`Obx` hanya bisa dipakai pada sebuah child yang memiliki update UI dari `observable`. Dan akan `error` bila implementasi `bukan` merupakan child yang memiliki update UI dari observable.

Link Github

Berikut merupakan link github yang akan digunakan pada modul pemrograman mobile:

[Link Github](#)

KEGIATAN PRAKTIKUM

A. Instalasi State Management

1. Lakukan instalasi State Management GetX pada project kalian dengan menambahkan dependencies di file `pubspec.yaml`
2. Kemudian lakukan save/pub get
3. Jalankan project seperti biasa

B. Refactoring

1. Lakukan refactoring folder project kalian dengan menerapkan architecture pattern, misalnya kalian menerapkan MVC pada project kalian maka buatlah folder sesuai pattern MVC tersebut.
 2. Masukkan file project modul 1 kalian sebelumnya ke dalam folder yang sudah dibuat.
 3. Opsi tambahan lain yaitu lakukan refactoring ke codingan kalian agar codingan lebih rapi dan mudah untuk dibaca.
-

RUBRIK PENILAIAN MODUL 2 MATERI

Bobot Penilaian Modul 2 Materi (20%)

Berhasil melakukan instalasi State Management	35
Berhasil melakukan refactoring folder sesuai architecture pattern yang dipakai	45
Melakukan refactoring pada codingan sehingga lebih rapi dan mudah dibaca	20
Total	100