**Getx**

1. **Package Install**
2. **MaterialApp replace With GetmaterialApp()**
3. **Get.snackBar()**
4. **Get.defaultdialog()**
5. **Get.BottomSheet()**

Getx Navigation

1. **Get.to(class Name);**
2. **Get.back();**
3. **Get.off():** Goto next screen but remove just the previous screen from the stack. In simple words, only one screen will be removed and you will be able to return back to – previous to previous screen.
4. **Get.offAll():** Goto next screen but remove all previous screens from the stack. In Simple words no back buttom. Same as **PushReplacement.**

**Named Route Navigation**

getPages: [

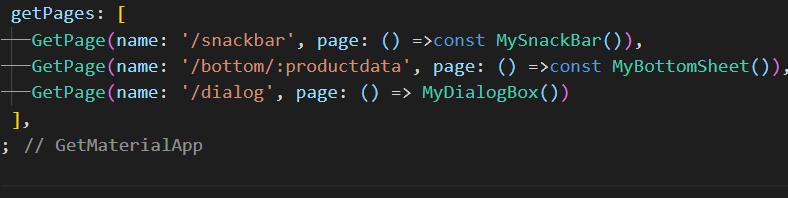
        GetPage(name: '/snackbar', page: () => MySnackBar()),

        GetPage(name: '/bottom', page: () => MyBottomSheet()),

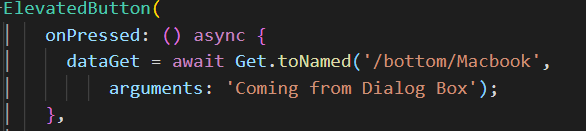
        GetPage(name: '/dialog', page: () => MyDialogBox())

      ],

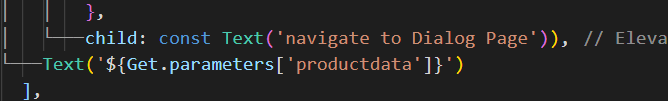
**Named Route Navigation With Parameters**



Pass data



Call Data



**Dependency Injection**

It is a technique of injecting instances of one class into another.

A basic approach to inject dependencies in flutter is through constructors.

1. Get.put() Makes the Dependency available to all the child Routes.

SO, in case we need to access the same instance on some other class, we can do that using Get.find().

Get.put(ControllerName)

By default, the dependencies will be deleted if the route using Get.put() is removed from the navigation stack. You may want to prevent this and keep the dependencies in memory for the entire app session. You can do that using permanent preoperty.

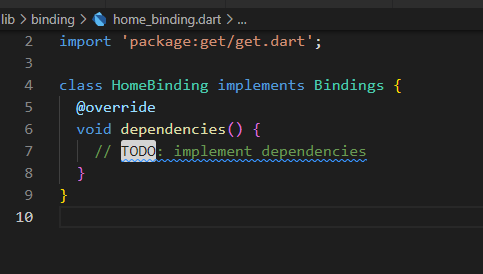
Get.put(ControllerName(), permanent: true);

**Bindings**

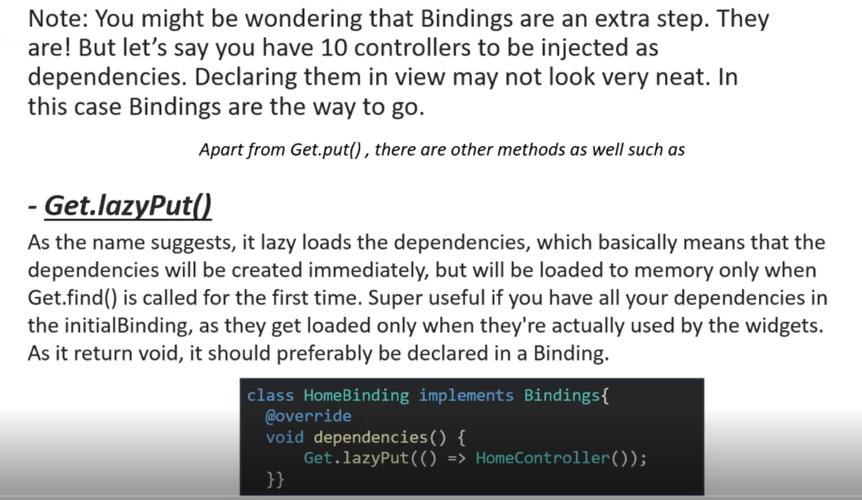
We are still declaring our dependencies in the view class. For organizing the code even further, we should separate them from the view using **Bindings.**

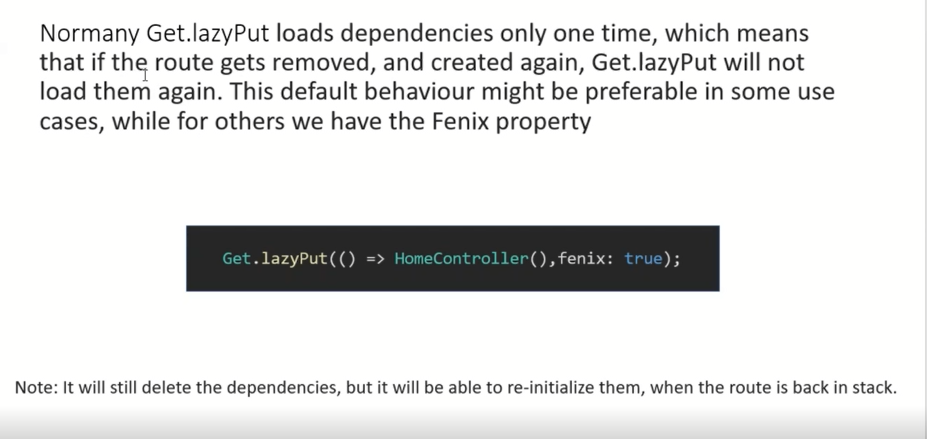
Bindings are classes where we can declare our dependencies and then *bind* them to routes.

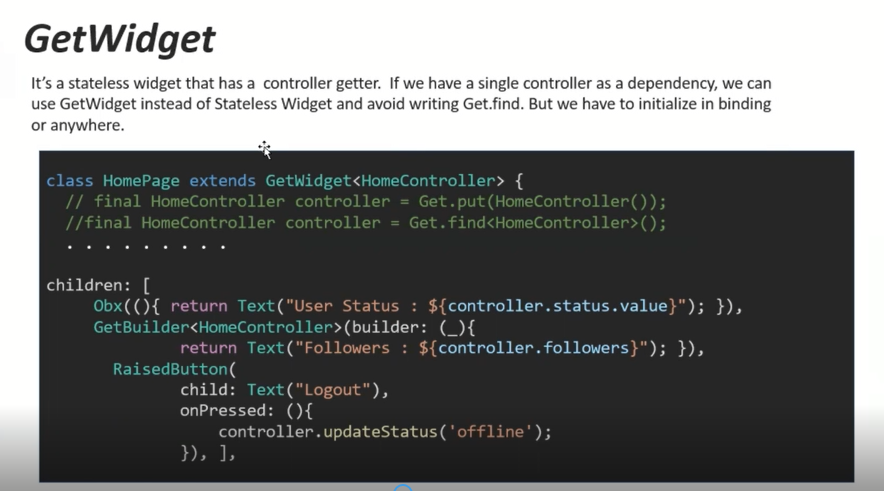
Create a new folder Bindings and inside that we start by creating a class **HomeBinding** that implements **Binding.**



Get.lazyPut()

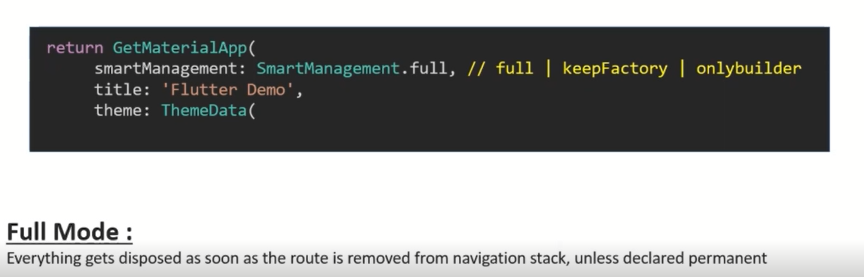


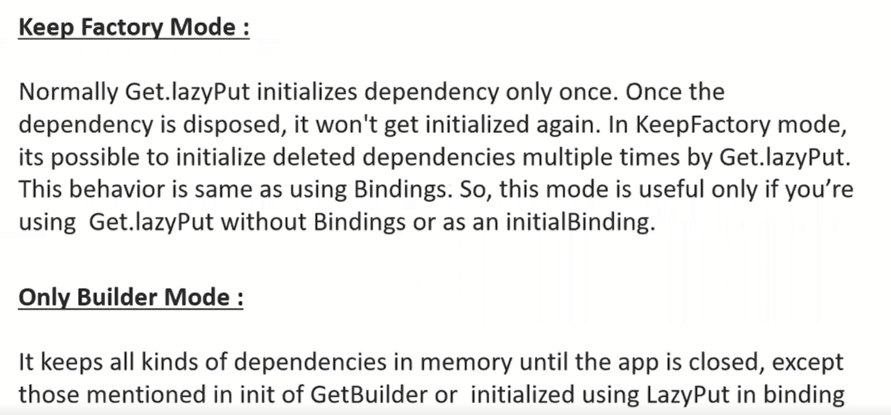


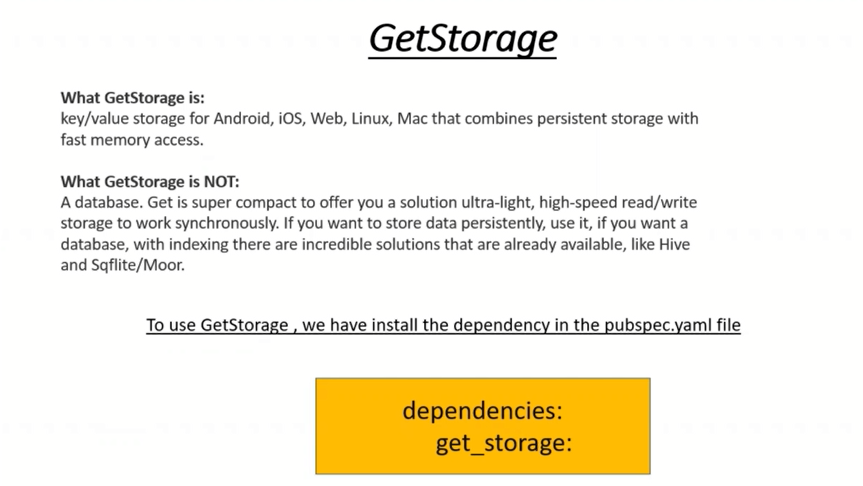


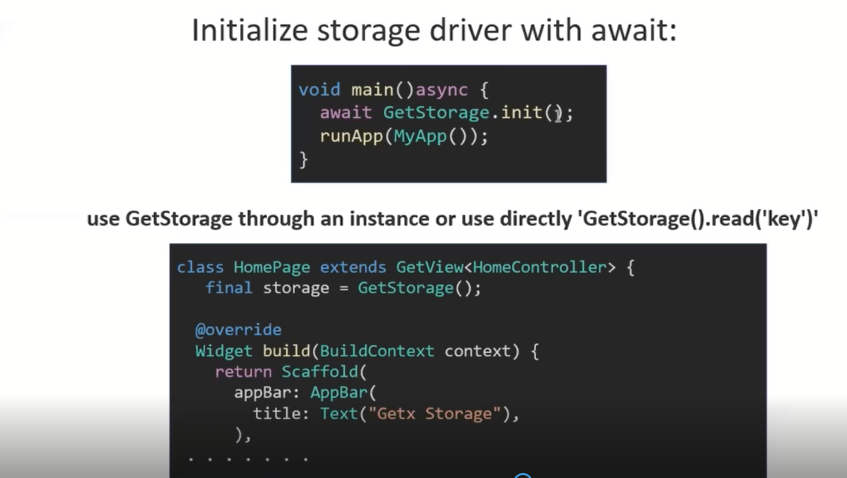
**Configuration**

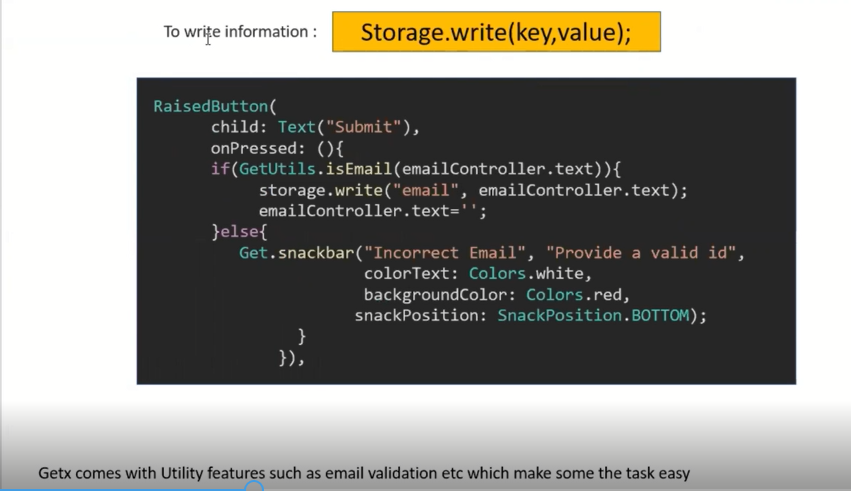
GetX provides something called Smart Management that lets us configure how dependencies behave in terms of memory management.











**Read Data**

