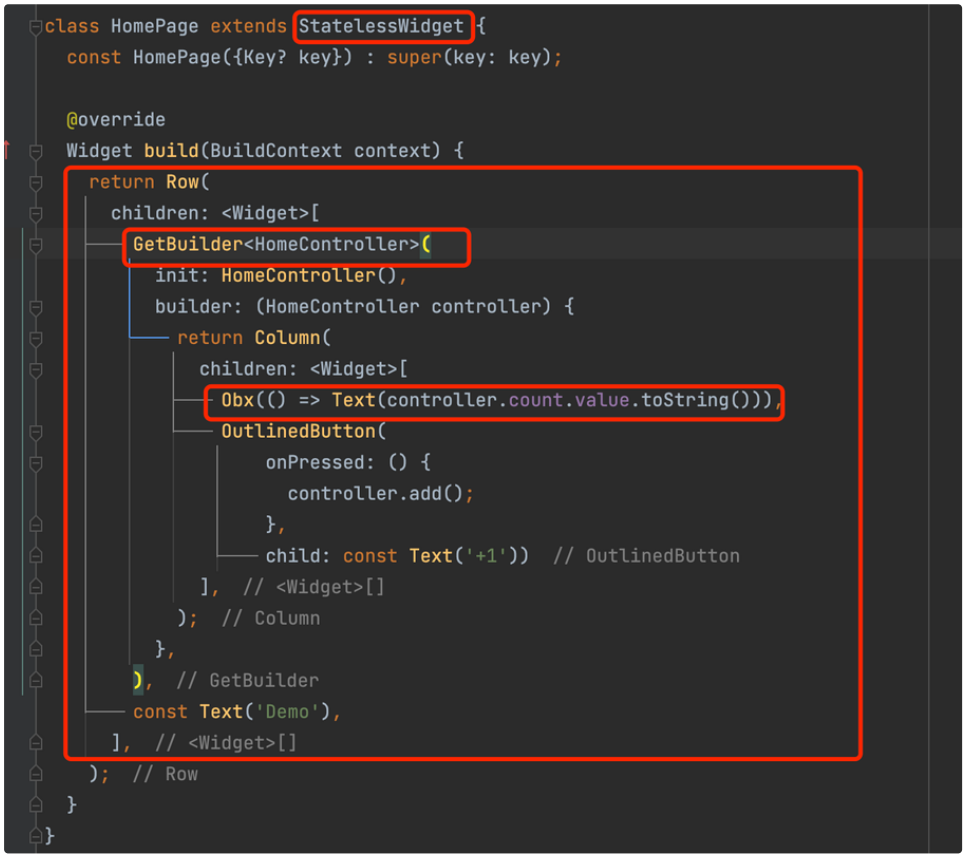
业务代码框架模版

模版参考代码： GitHub - chejdj/flutter_project_template: Template for Flutter project

项目使用框架： GetX GitHub - jonataslaw/getx: Open screens/snackbars/dialogs/bottomSheets without context, manage states and inject dependencies easily with Get.

需要注意几个问题：

- 

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      children: <Widget>[
        GetBuilder<HomeController>(
          init: HomeController(),
          builder: (HomeController controller) {
            return Column(
              children: <Widget>[
                Obx(() => Text(controller.count.value.toString())),
                OutlinedButton(
                    onPressed: () {
                      controller.add();
                    },
                    child: const Text('+1'))  // OutlinedButton
              ],  // <Widget>[]
            );  // Column
          },
        ),  // GetBuilder
        const Text('Demo'),
      ],  // <Widget>[]
    );  // Row
  }
}
```

  页面使用 StatelessWidget，再也没有必要使用StatefullWidget
- 使用GetBuilder 进行 Controller 和 UI 视图的链接，切忌区分页面中可变以及不可变的元素，不要一味的把GetBuilder放到最顶层(hint: GetBuilder还会自动回收Controller 当页面销毁的时候)
- 当使用到obs 变量的时候，需要使用Obx进行包裹，实现局部状态更新

```dart
class HomeController extends GetxController {
  Rx<User> user = Rx<User>(User());
  RxInt count = 0.obs;

  @override
  void onInit() {
    super.onInit();
    print('widget alloct memory call this');
    fetchUser();
  }

  @override
  void onReady() {
    super.onReady();
    print('first frame call this');
  }

  @override
  void onClose() {
    super.onClose();
    print('onClose');
  }

  void add() {
    count++;
  }

  Future<void> fetchUser() async {
    HomeApi.fetchUserInfo().then((User value) {
      user.call(value);
    });
```

- 使用Controller 注意 Controller的生命周期, onInit, onReady, onClose, 进行一些初始化工作，以及资源释放
- 当涉及到跨组件共享数据的时候，可以使用把Controller持久到全局对象池中，但是需要注意这种情况，**需要自己去调用delete 在合适时机回收对象**

```dart
Get.put(HomeController(),permanent: true);
```

- GetX 提供的工具类比较丰富，同时像状态管理，路由管理，依赖管理，他们之间也没有什么耦合。因为我们项目当中有比较成熟路由这一块可以先使用项目的路由管理。使用其他新的组件的时候，需要多看看实现，GetX的文档相对来说还是差一些。