[Flutter]环境判断

风浅月明

方式一（推荐）

常量kReleaseMode，它会根据你的应用是以什么模式编译的来获取值。bool.fromEnvironment会从Dart编译时的环境变量中获取值。对于'dart.vm.product'这个特定的环境变量，它是由Dart VM设置的，用来标明当前是否在产品环境下运行。

当应用以Release模式编译时（例如运行flutter build apk或flutter build ios），kReleaseMode会被设置为true。

当应用在Debug模式或Profile模式下运行时，kReleaseMode会被设置为false。

kReleaseMode是foundation库的一部分，这意味着你不需要手动定义它，可以直接使用。这个变量与其他几个常量一起，帮助开发者编写依赖于构建模式的代码：

kDebugMode: 当应用在Debug模式下运行时为true。

kProfileMode: 当应用在Profile模式下运行时为true。

kReleaseMode: 当应用在Release模式下运行时为true。

```
// const bool kReleaseMode =
bool.fromEnvironment('dart.vm.product');
if (kReleaseMode) {
  print("dart.vm.product-现在是release环境.");
} else {
```

```
  print("dart.vm.product-现在是debug环境.");
}
```

1.在Main.dart选择Start Debugging运行

还用 Android Studio 和 Xcode 运行一样默认会是 debug 环境

打印结果：

I/flutter (23746): dart.vm.product-现在是 debug 环境.

2.使用flutter run运行

```
$ flutter run
```

打印结果：

I/flutter (24584): dart.vm.product-现在是 debug 环境.

3.使用flutter run --debug运行

```
$ flutter run --debug
```

打印结果：

I/flutter (30485): dart.vm.product-现在是 debug 环境.

4.在终端使用flutter run --release运行

```
$ flutter run --release
```

打印结果：

I/flutter (26343): dart.vm.product-现在是 release 环境.

5.使用flutter build apk打包安装

```
$ flutter build apk
```

提示：

dart.vm.product-现在是 release 环境.

6.使用flutter build apk --debug打包安装

```
$ flutter build apk --debug
```

提示：

dart.vm.product-现在是debug环境.

7.打包debug或release的ipa用Xcode打开项目操作

在Xcode中，默认情况下运行或构建应用会使用Debug配置，这意味着如果你直接通过Xcode的运行按钮（通常是顶部左侧的一个播放按钮）启动应用，它将默认使用Debug模式。这也意味着你的kReleaseMode将会是false。

如果你想要通过Xcode显式地打包一个Debug模式的ipa文件，你可以遵循以下步骤：

打开你的Flutter项目中的ios文件夹。你可以在终端使用open ios/Runner.xcworkspace命令来打开Xcode项目，或者直接在Finder中找到Runner.xcworkspace文件并双击打开。

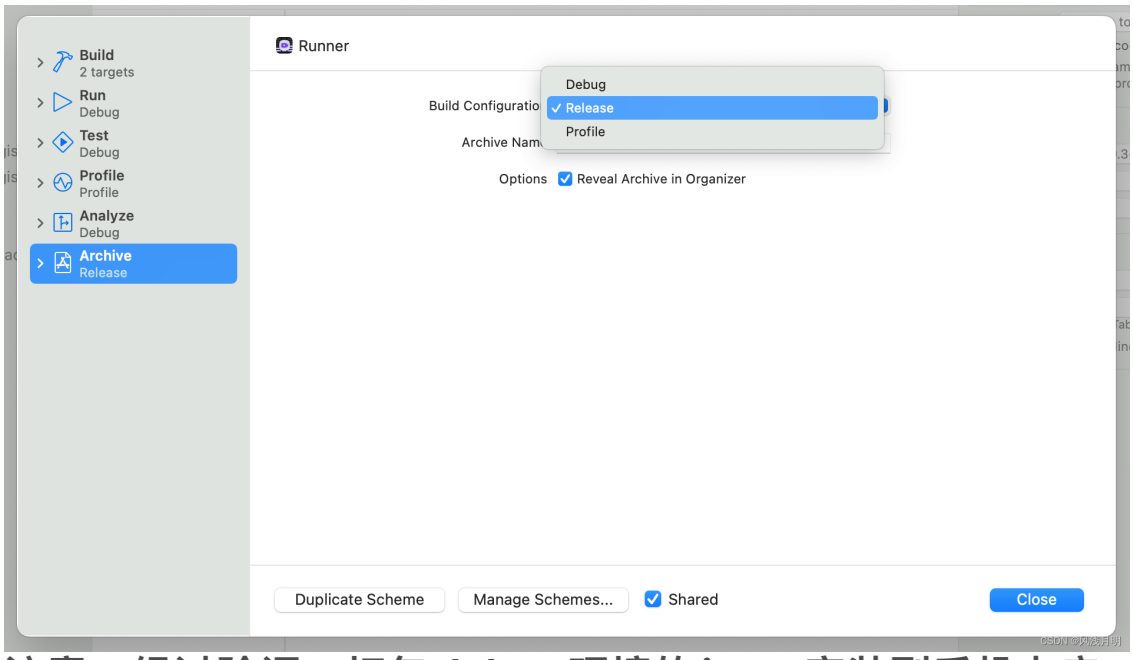
确保你的设备或者是一个有效的模拟器是当前选中的目标设备。

前往Xcode的顶部菜单栏，选择Product > Scheme > Edit Scheme。

在左侧菜单中选择Archive，然后在右侧的Build Configuration中选择Debug。(左侧选择Run的话，就是修改运行后的安装包环境)

关闭Scheme编辑器，然后去到Product > Archive来创建一

个新的归档。 注意：归档操作通常用于准备 Release 模式的构建，但是你可以改变 Scheme 设置来创建 Debug 模式的归档。
一旦归档完成，Xcode 的 Organizer 窗口会打开，显示你的新归档。
在 Organizer 中，选择你的归档，然后点击右边的 Export 按钮来导出 ipa 文件。
在导出流程中，确保选择正确的导出选项，比如 Development 来保持 Debug 模式。
在这个过程完成后，你将会得到一个 Debug 模式的 ipa 文件，它包含了调试符号和不是针对性能优化的编译。你可以将这个 ipa 文件安装到你的设备上，用于测试目的。记住 Debug 模式的构建并不适合发布到 App Store 或其他分发渠道。
最后，不管是通过 Xcode 还是通过命令行工具，kReleaseMode 的值始终由构建配置来决定，当使用 Release 配置构建时，kReleaseMode 为 true；当使用 Debug 配置构建时，kReleaseMode 为 false。



注意：经过验证，打包debug环境的ipa，安装到手机上启动不了，打包release安装没问题。

方式二

若你不想用kReleaseMode判断环境，你还可以用“ENV”甚至自定义一个变量来判断环境。

使用String.fromEnvironment('ENV')判断环境时，若编译时没用--dart-define指定环境变量（flutter run --dart-define=ENV=DEBUG 或这样 flutter build apk --dart-define=ENV=RELEASE），默认会返回空字符串。

在Flutter开发中，--dart-define是一个命令行标志，它允许开发者在编译时向 Dart 代码注入环境变量或配置数据。这使得你可以在不同的开发环境中（如开发、测试、生产）指定不同的变量值，而无需更改源代码。

下面的代码中，我们通过判断environment是否等于'DEBUG'，让环境默认为Release环境。

```
// 在你的代码中，确保 environment 声明前面有 const 关键字。如果没有 const，则不会在编译时查找环境变量的值，而是会默认为空字符串。
const String environment = String.fromEnvironment('ENV');
if (environment == 'DEBUG') {
  print("ENV-现在是debug环境. environment = $environment");
} else {
  print("ENV-现在是release环境. environment = $environment");
}
```

这种劣势也比较明显，因为需要编译时手动指定环境，开发时忘记指定--dart-define=ENV=DEBUG时，就容易误入release环境。比如你在release环境中有埋点，误入后就会制造出很多垃圾数据。

1.使用flutter run --dart-define=ENV=DEBUG运行

```
$ flutter run --dart-define=ENV=DEBUG
// 或者
$ flutter run --dart-define="ENV=DEBUG"
```

打印结果：

```
I/flutter (29771): ENV-现在是debug环境. environment = DEBUG
```

2.使用flutter build apk --dart-define=ENV=RELEASE打包安装

```
$ flutter build apk --dart-define=ENV=RELEASE
```

提示：

ENV-现在是release环境. environment = RELEASE