Flutter（二十三）编译模式

AlanGe

Flutter 编译模式

在 Android 和 iOS 中，应用程序运行分为 debug 和 release 模式，分别对应调试阶段和发布阶段；

在 Flutter 中，应用程序分为以下三种模式

1.debug

2.profile　　profile 侧面，外形

3.release

下面我们就聊一下三种模式的区别和应用；

一. Flutter 编译模式

1.1. debug 模式

在 **Debug 模式**下，app 可以被安装在真机、模拟器、仿真器上进行调试。

Debug 模式有如下特点：

- 断言是开启的（Assertions）

- 服务扩展是开启的（Service extension）
    - 这个可以从 runApp 的源码查看　查看runApp的代码
    - runApp -> WidgetsFlutterBinding -> initServiceExtensions
- 开启调试，类似于 DevTools 的工具可以连接到应用程序的进程中
- 针对快速开发和运行周期进行了编译优化（但不是针对执行速度、二进制文件大小或者部署）
    - 比如 Dart 是 JIT 模式（Just In Time，即时编译，也可以理解成 边运行边编译）

默认情况下，运行 `flutter run` 会使用 Debug 模式，点击 Android Studio **run 按钮**，也是 debug 模式

```
Launching lib/main.dart on iPhone 11 in debug mode...
```

下面的情况会出现在 **Debug 模式**下：

- 热重载（Hot Reload）功能**仅能**在调试模式下运行；
- 仿真器和模拟器**仅能**在调试模式下运行；
- 在 debug 模式下，应用可能会出现掉帧或者卡顿现象；

1.2. release 模式

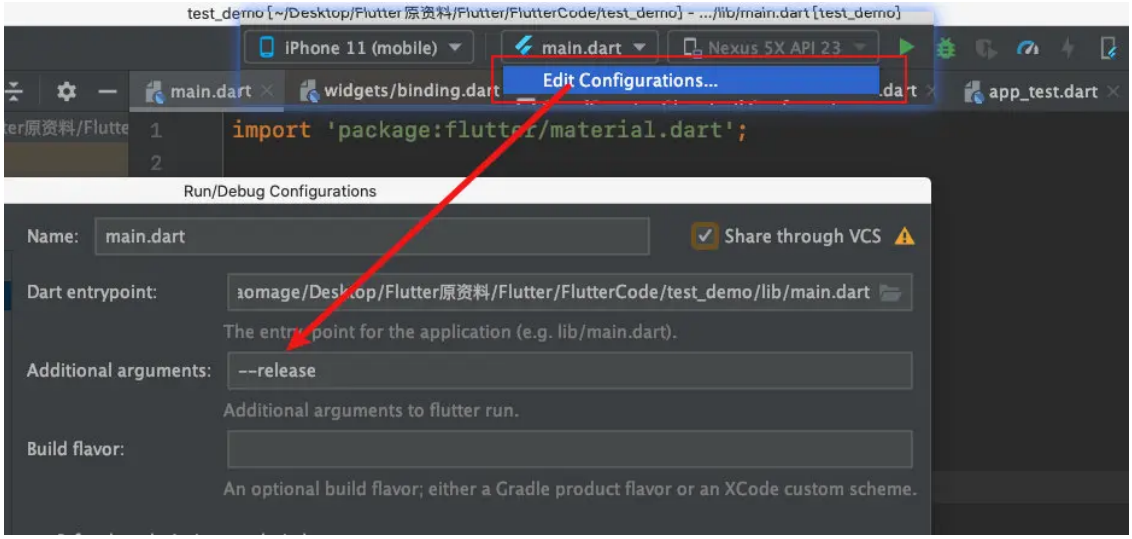
当我们要发布应用程序时，总是希望最大化的优化性能和应用程序所占据的空间。

在 **Release 模式**下是不支持模拟器和仿真器的，只能在真机上运行。

Release 模式有如下特点：

- 断言是无效的
- 服务扩展是无效的
- debugging 是无效的
- 编译针对快速启动、快速执行和小的 package 的大小进行了优化
    - 比如 Dart 是 AOT 模式（Ahead Of Time，预先编译）

`flutter run --release` 命令会使用 Release 模式来进行编译，也可以给Android Studio进行配置：



如果继续运行在模拟器上：

```
Release mode is not supported for emulators.
```

1.3. profile 模式

profile 模式和 **release 模式**类似，但是会保留一些信息方便我们对性能进行检测。

profile 模式有如下特点：

- 保留了一些扩展是开启的；
- DevTools 的工具可以连接到应用程序的进程中；

Profile 模式最重要的作用就是可以利用 DevTools 来测试应用的性能；

二. 开发中模式区分

在开发中，我们可能想要对 debug 和 release 模式进行区分，根据不同的模式进行不同的相关设置：

- 比如网络请求的 baseURL

如何进行区分呢？常见的有两种方式：

- 通过 assert 断言，因为在 release 模式下断言是无效的
- 通过 kReleaseMode 常量来区分

通过断言 assert 来区分：

- 因为 assert 要求我们必须传入一个 bool 值，所以我们使用了一个立即执行函数

```
String baseURL = "production baseURL";
assert(() {
  baseURL = "development baseURL";
  return true;
}());
```

```
const MyHomePage({super.key, required this.title}):assert(title == 'abc
', 'title isnot abc');
```

通过 kReleaseMode 常量来区分

```
String baseURL = kReleaseMode ? "production baseURL":
"development baseURL";
```

当然，上面只是针对 baseURL 来进行了区分，开发中如果有

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

多个属性需要区分呢?

- 可以封装一个 Config 的类，通过 InheritedWidget 来进行共享即可
- 大家可以利用之前学习过的 InheritedWidget 来自行封装