> 想学Flutter，就请关注这个专栏
>
> Flutter系列（一）——详细介绍
>
> Flutter系列（二）——与React Native进行对比
>
> Flutter系列（三）——环境搭建（Windows）
>
> Flutter系列（四）——HelloWorld）

文档归档： https://github.com/yang0range/flutterfile

创建的第一个Demo

上一篇文章，详细的介绍了Flutter的环境搭建，搭建完成之后，自然迫不及待看看如何创建我们的第一个Demo。

打开**Android Studio**

可以看到，可以看见有四个选项。

那么这四个选项有什么区别呢？

Flutter Application
Flutter标准引用程序

Flutter Plugin
Flutter封装的Native工程的基础插件

Flutter Package
纯**Dart**库工程

Flutter Modue
作为已有工程的一个**Module**

这里，我们需要创建一个**Flutter Application**项目。

工程目录结构

这里，我们首先可以看到三个比较重要的目录，分别是**android**，**ios还有lib**。

android
顾名思义，就是写Android平台相关代码的地方。

ios
和上面类似，这就就是写ios平台相关代码的地方。

lib
这里才是我们真正写flutter相关代码的地方。

这里还有一个十分重要的文件，

pubspec.yaml
这个文件是我们**flutter**的配置文件，比如三方的依赖都在这里，最重要的是这里还要写一些资源文件，比如图片等等，后面我们会详细介绍。

从上面目录结构大致可以理解为，整理的**flutter**的工程结构为。

运行官方Demo

先把官方Demo运行起来。

运行不起来

我们第一次运行官方Demo的时候，可能会有运行不来，一直卡在
Running Gradle task 'assembleDebug'...
的情况，这里是因为Gradle的Maven仓库在国外，因为众所周知的问题，无法加载。

解决办法

这里，我们就可以使用阿里云的镜像地址，来解决这个问题。 打开如下目录，修改的地方。

也可以直接复制如下代码。 ``` buildscript {
ext.kotlin_version = '1.3.50'
repositories {
// 这里做了修改，使用国内阿里的代理
// google()
// jcenter() maven { url 'https://maven.aliyun.com/repository/google' }
maven { url 'https://maven.aliyun.com/repository/jcenter' }
maven { url 'http://maven.aliyun.com/nexus/content/groups/public' }
}
dependencies {
classpath 'com.android.tools.build:gradle:3.5.0'
classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
}
}

allprojects {
repositories {
//修改的地方
//google()
//jcenter() maven { url 'https://maven.aliyun.com/repository/google' }
maven { url 'https://maven.aliyun.com/repository/jcenter' }
maven { url 'http://maven.aliyun.com/nexus/content/groups/public' }
}} 接着打开`flutter.gradle`文件，路径在 `**..\fluttersdk\flutter\packages\flutter_tools\gradle**`下，修改内容 buildscript { repositories { // 这里做了修改，使用国内阿里的代理 // google() // jcenter() maven { url 'https://maven.aliyun.com/repository/google' } maven { url 'https://maven.aliyun.com/repository/jcenter' } maven { url 'http://maven.aliyun.com/nexus/content/groups/public' } } dependencies { classpath 'com.android.tools.build:gradle:3.5.0' } } ``` 再重新运行，就成功了！

官方Demo结构

打开**main.dart**文件，就是运动Demo的代码了。

根据代码，我们可以画出这个Demo的结构如下图。

官方Demo中几个重要的内容

下面我们着重依次介绍一下。

void main() => runApp(MyApp());

入口函数，这里用的是Dart语法当中的箭头函数，这里和Kotlin的用法十分相像。

我们可以看到这里的**MyApp()**继承的是**StatelessWidget**而**StatelessWidget**继承的是**Widget**

可以说在flutter当中，一切皆为widget

这里的 **runApp**起到了一个全局更新的作用，一般flutter启动时调用后不会再调用

MyApp

这里的Myapp返回的是一个**MaterialApp**相信了解Android的小伙伴对这个不会陌生，这个就是让这个Flutter保持一个M**Material**的UI风格。

当然，也有IOS的风格，这里我们可以使用**CupertinoApp**。

StatelessWidget

无中间状态变化的widget，初始状态设置以后就不可再变化， 用于不需要维护组件状态的场景， createElement()创建StatelessElement对象，一个StatelessWidget对应一个StatelessElement。

可以看到MyApp就是继承自StatelessWidget。

StatefulWidget

存在中间状态变化的widget，createElement()创建StatelfulElement对象 ，createState()创建State对象(可能调用多次)，createState()创建State对象(可能调用多次)。

MyHomePage因为有点击的count++的导致UI变化，所以继承自StatefulWidget。

State

State是一个状态对象，<>里面是表示该状态是跟谁绑定的。

State有两个作用 1.在修改状态就在这个类里编写，Weidget的时候可以同步的读取。

2.当状态有所改变的时候，调用State.setState()同时去刷新Weidget。

State.setState()

将子树作StatefulWidget的一个子Widget，并创建对应的State类实例，通过调用State.setState()触发子树的刷新。

在Demo当中，可以看到_MyHomePageState就是继承自State并且通过State.setState()这个方法局部刷新UI。

最后

以上就是关于整个官方Demo的一个较为详细的介绍，也是我们接触的第一个Flutter项目，接下来我们就了解一下什么是Dart语言，看看Dart语言有哪些特点，为什么Flutter要使用Dart语言。

Flutter已经是Top20的软件库，通过接下来的一系列的文章，希望我和大家一起来学习Flutter，一起进步，一起有所收获，掌握未来技术主流的主动权！

有什么好的建议，意见，想法欢迎给我留言！

欢迎关注公共号

关注公众号会有更多收获！

动动小手指点赞，收藏，关注一键三连走一波吧！