# 15.2 Flutter APP代码结构

我们先来创建一个全新的Flutter工程，命名为“github\_client\_app”；创建新工程的步骤视读者使用的编辑器而定，都比较简单，在此不再赘述。创建完成后，工程结构如下：

github\_client\_app  
├── android  
├── ios  
├── lib  
└── test

由于我们需要使用外部图片和Icon资源，所以我们在项目根目录下分别创建“imgs”和“fonts”文件夹，前者用于保存图片，后者用于保存Icon文件。关于图片和Icon，读者可以参考第三章中相应的内容。

由于在网络数据传输和持久化时，我们需要通过Json来传输、保存数据；但是在应用开发时我们又需要将Json转成Dart Model类，现在我们使用在第十一章中“Json转Model”小节中介绍的方案，所以，我们需要在根目录下再创建一个用于保存Json文件的“jsons”文件夹。

多语言支持我们使用第十三章“国际化”中介绍的方案，所以还需要在根目录下创建一个“i10n”文件夹，用于保存各国语言对应的arb文件。

现在工程目录变为：

github\_client\_app  
├── android  
├── fonts  
├── i10n-arb  
├── imgs  
├── ios  
├── jsons  
├── lib  
└── test

由于我们的Dart代码都在“lib”文件夹下，笔者根据技术选型和经验在lib文件下创建了如下目录：

lib  
├── common  
├── i10n  
├── models  
├── states  
├── routes  
└── widgets

|  |  |
| --- | --- |
| 文件夹 | 作用 |
| common | 一些工具类，如通用方法类、网络接口类、保存全局变量的静态类等 |
| i10n | 国际化相关的类都在此目录下 |
| models | Json文件对应的Dart Model类会在此目录下 |
| states | 保存APP中需要跨组件共享的状态类 |
| routes | 存放所有路由页面类 |
| widgets | APP内封装的一些Widget组件都在该目录下 |

注意，使用不同的框架或技术选型会对代码有不同的组织方式，因此，本节介绍的代码组织结构并不是固定或者“最佳”的，在实战中，读者可以自己根据情况调整源码结构。但是无论采取何种源码组织结构，但清晰和解耦是一个通用原则，我们应该让自己的代码结构清晰，以便交流和维护。