第 1 章　初识 Flutter

1.1 Flutter 简介

随着移动互联网兴起，移动应用开发也逐渐兴起，不过传统的移动应用开发需要同时兼顾多端开发，这不仅大大降低了项目开发的效率，也不能适应移动应用高速迭代的需求。为了提升开发效率，同时也为了节约多端开发带来的人力成本，不少公司和开发者一直都在寻找一种可以高效开发的移动跨平台开发方案。

纵观移动跨平台技术的发展历史，大体经历了三个时代，一是使用原生内置浏览器加载网页的 Hybrid 方案，代表技术有 Cordova、Ionic 和微信小程序；二是使用 JavaScript 语言开发，再使用原生平台组件渲染的原生渲染方案，代表技术有 React Native、Weex 和快应用；三是使用自带的渲染引擎和组件实现的跨平台渲染方案，代表技术是 QT Mobile 和 Flutter，此种方案屏蔽了底层操作系统的差异，真正实现了软件应用层的跨平台开发。

抛开成熟且体验较差的 Hybrid 技术，当前市面上流行的移动跨平台方案主要指的是 React Native 和 Flutter。在技术实现方面，React Native 使用 JavaScript 语言进行开发，然后再使用 JavaScriptCore 将 JavaScript 代码解析成原生移动平台组件后再执行界面渲染。Flutter 则使用 Skia 渲染引擎（3.0 之后的版本使用的是 Impller 渲染引擎），它是一个跨平台的 2D 渲染引擎，Google Chrome 浏览器和 Android 的 2D 渲染都是使用 Skia 引擎来执行渲染的，因此渲染效率上相比 JavaScriptCore 方式来说要高很多。最新版本的 Flutter 已经使用了优化后的 Impeller 渲染器，其渲染性能更是得到了大幅的提升。

不过，在技术选型方面，究竟选择哪一种跨平台技术，还需要从开发效率、渲染性能、维护成本和社区生态等多个方面进行评估。总体来说，Flutter 在开发效率和渲染性能方面是目前最好的移动跨平台方案，并且，Flutter 在开启了对 Web 和桌面应用的支持后，真正成为横跨移动、Web 和桌面开发的跨平台技术方案。

1.1.1 Flutter 诞生历史

作为由全球知名软件厂商 Google 公司主导的跨平台技术方案，Flutter 自从 2017 年 5 月首个正式版本被发布以来，受到了广大开发者和企业的追捧，被大量应用在商业项目开发中。Flutter 在 1.5.0 版本新增了对 Web 环境以及在 2.0.0 版本新增了对 Windows、macOS 等桌面环境的支持后，已经真正意义上实现了跨平台稳定运行的愿景。

目前，经过近 6 年的迭代，Flutter 已经发布了 3.7.8 正式版，解决了之前遗留的性能问题和平台适配问题。并且，虽然 Flutter 已经发布多年，但保持着每个月更新一个版本的节奏。

如果大家想要体验最新版本的 Flutter，可以打开 Flutter 托管在 GitHub 上的地址来获取最新的源码，如图 1-1 所示。

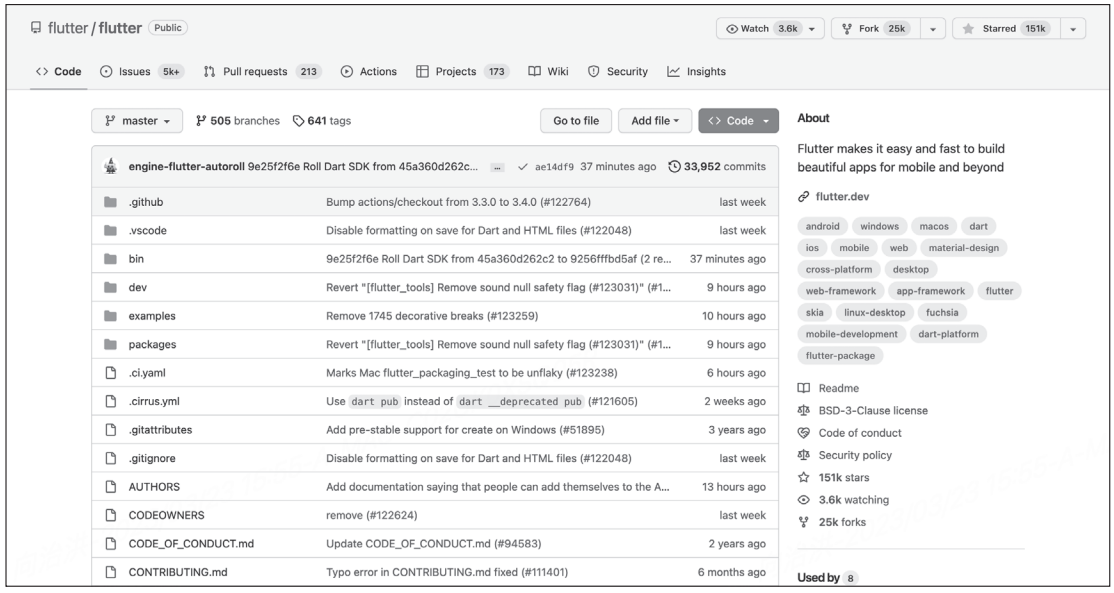

图 1-1　获取 GitHub 上 Flutter 源码

可以看到，相比其他跨平台技术方案，Flutter 是目前获得关注最多的跨平台方案，其受关注的程度更是直追前端框架 React、Vue。

1.1.2 Flutter 优势

作为由 Google 公司推出的开源跨平台技术框架，Flutter 主打的是跨平台、高保真和高性能。作为时下最流行的跨平台技术框架，Flutter 具备如下一些优点。

（1）跨平台特性。

Flutter 支持运行在 iOS、Android、Windows、macOS、Linux 和 Fuchsia 等操作系统上，真正做到了“一次编写，处处运行”，节约了人力和开发成本。

（2）高性能。

Flutter 使用全新的 Impeller 渲染引擎来绘制视图，基本可以保证应用在运行时达到 60 帧 / 秒，从体验上来说，和原生技术开发的应用没有太大的区别。

（3）响应式框架。

Flutter 提供的响应式框架和一系列基础组件，可以帮助开发者快速地构建用户界面。并且，响应式框架的另一个优点是可以完美适配不同的屏幕分辨率，解决不同分辨率设备的体验问题。

（4）热重载。

在传统的移动原生应用开发过程中，当出现问题时，往往需要修改缺陷然后重新运行。而 Flutter 提供的热重载功能，可以帮助开发者无须重新启动应用，即可完成测试、用户界面构建和错误修复。

之所以如此高效，是因为 Flutter 可以将更新后的源代码文件注入正在运行的 Dart 虚拟机中，在虚拟机使用新字段和函数更新后，Flutter 框架会自动重新构建组件树，并自动刷新界面。

（5）开发效率高。

Flutter 使用 Dart 语言进行开发，该语言的语法特性对前端开发者非常友好。同时，Flutter 在开发阶段采用 JIT（即时编译）模式，避免了每次代码改动都要进行编译，极大地节省了开发时间。而在发布阶段，Flutter 采用在 AOT（运行前编译）模式，保证了应用的体验和运行性能。

1.1.3　Flutter 版本

目前，Flutter 对外发布的渠道主要有 Master、Dev、Beta 和 Stable 四种类型。并且，这些对外渠道的稳定性依次提高，但新特性却逐渐减少。说明如下：

Master：Master 渠道的代码是最新的，包含了最新的试验性功能和特性。不过，Master 渠道的代码没有经过严格的测试，可能会出现各种各样的问题。

Dev：Dev 渠道的代码是经过 Google 公司内部测试后的版本，相比 Master 渠道缺陷会更少，但是这并不意味着不存在缺陷。因为 Dev 渠道的测试都是一些最基础的测试，一旦发现有严重的缺陷，Dev 渠道会被直接废弃。

Beta：Beta 渠道是经过 codelabs 测试的 Dev 渠道，是运行稳定的 Dev 渠道，因此 Beta 渠道不会有严重的缺陷问题，在稳定性方面是最接近 Stable 的渠道。

Stable：Stable 渠道是官方发布的最稳定的版本，通常是从 Beta 渠道选取出来的，也是官方推荐的使用版本。

当然，如果本地已经安装了某个渠道版本，可以使用下面的命令进行版本的切换，命令如下：

```
flutter channel
flutter channel channel-name          // channel-name 为渠道
```

1.2 Flutter 框架

与原生 Android、iOS 系统一样，Flutter 框架也是一个分层的架构，每一层都建立在前一层之上，并且上层比下层的使用频率更高，是直接面向开发者的，官方给出的 Flutter 框架的架构如图 1-2 所示。

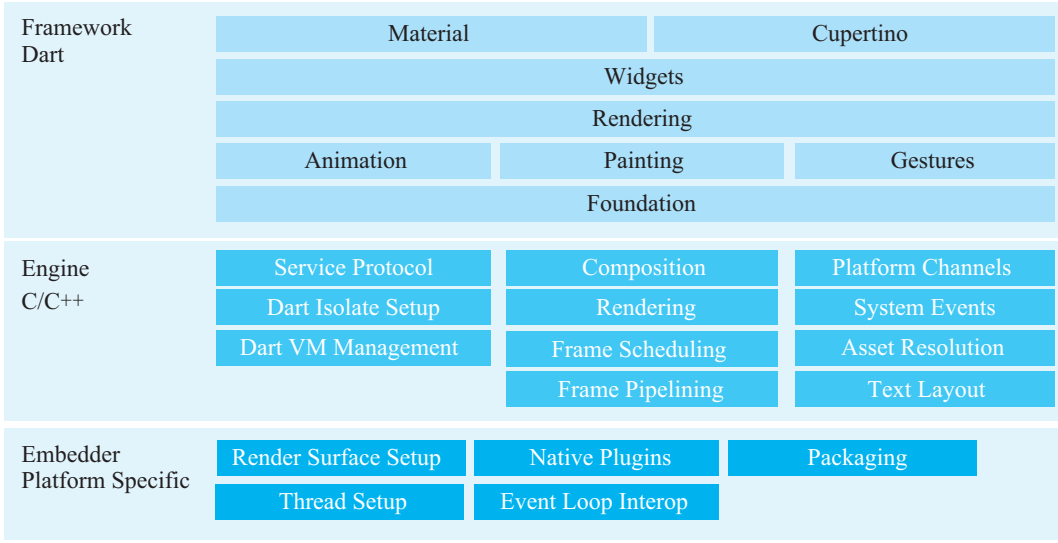

图 1-2　Flutter 架构示意图

可以看到，Flutter 框架从上到下可以分为 Framework、Engine 和 Embedder 三层。

其中，Framework 表示框架层，主要由纯 Dart 编写的基础组件库构成，提供底层 UI 库、动画、手势及绘制等功能，是普通开发者使用频率最高的一层。Embedder 表示嵌入层，主要作用是提供渲染 Surface 设置、线程管理和提供操作系统适配等。

1.2.1 Flutter Framework

Framework 表示框架层，是一个由 Dart 实现的软件开发工具包，实现了一套自底向上的基础库，用于处理动画、绘图和手势等。下面按照自底向上的顺序，对 Framework 层进行说明。

（1）底下两层：Foundation 和 Animation、Painting 和 Gestures 等合并为一个 Dart UI 层，对应 Flutter 中的 dart:ui 包，是 Flutter Engine 暴露的底层 UI 库，主要用于提供动画、手势及绘制等能力。

（2）Rendering 层：抽象的布局层，它依赖于 Dart 的 UI 层，渲染层会构建一棵由可

渲染对象组成的渲染树，当动态更新这些对象时，渲染树会找出变化的部分，然后再更新渲染。

（3）Widgets 层：Flutter 提供的一套基础组件库，在基础组件库之上，Flutter 还提供了 Material 和 Cupertino 两种视觉风格的组件库，用来适配 Android 和 iOS 的设计规范。

1.2.2　Flutter Engine

Engine 表示引擎层，是一个由 C/C++ 实现的软件开发工具包，主要由 Skia 引擎、Dart 运行时、文字排版引擎和独立虚拟机构成。正是因为独立虚拟机的存在，Flutter 才能运行实现跨平台运行。

1.2.3　Flutter Embedder

Embedder 表示嵌入层，又被称为操作系统适配层，通过该层能够将 Flutter 嵌入到不同的操作系统平台，主要负责渲染 Surface 设置、线程管理和提供操作系统适配等工作。

为了适配不同的操作系统，嵌入层采用了平台语言进行编写，即 Android 使用的是 Java 和 C++，iOS 和 macOS 使用的是 Objective-C，Windows 和 Linux 则使用的 C++。正是由于 Embedder 层的存在，Flutter 才具备了跨平台运行的能力。目前，Flutter 已经适配了常见的操作系统平台，如果需要支持一些新的操作系统平台，则可以针对该平台编写一个嵌入层即可。