第1篇　Dart基础

第 1 章

Dart 语言简介

Dart 诞生于 2011 年 10 月 10 日，谷歌 Dart 语言项目的领导人 Lars Bak 在丹麦举行的 Goto 会议上宣布，Dart 是一种“结构化的 Web 编程”语言，Dart 编程语言在所有现代的浏览器和环境中提供高性能。

Dart 是谷歌开发的计算机编程语言，后来被 ECMA（ECMA-408）认定为标准。它被用于 Web、服务器、移动应用和物联网等领域的开发。它是宽松开源许可证（修改的 BSD 证书）下的开源软件。

Dart 有以下三个方向的用途，每一个方向，都有相应的 SDK。Dart 语言可以创建移动应用、Web 应用，以及 Command-line 应用等，如图 1-1 所示。

图 1-1　Dart 支持的平台

1.1　移动端开发

Dart 在移动端上的应用离不开 Flutter 技术。Flutter 是谷歌的移动 UI 框架，可以快速在 iOS 和 Android 上构建高质量的原生用户界面。Flutter 可以与现有的代码一起工作。在全世界，Flutter 正在被越来越多的开发者和组织使用，并且 Flutter 是完全免费、开源的。简单来说，Flutter 是一款移动应用程序 SDK，包含框架、控件和一些工具，可以用一套代码同时构建 Android 和 iOS 应用，并且其性能可以达到与原生应用一样的水源。

Flutter 采用 Dart 的原因很多，单纯从技术层面分析如下：

- Dart 是 AOT(Ahead Of Time)编译的,可编译成快速、可预测的本地代码,Flutter 几乎可以使用 Dart 编写;
- Dart 也可以 JIT(Just In Time)编译,开发周期快;
- Dart 可以更轻松地创建以 60fps 运行的流畅动画和转场;
- Dart 使 Flutter 不需要单独的声明式布局语言;
- Dart 容易学习,具有静态和动态语言用户都熟悉的特性。

Dart 最初设计是为了取代 JavaScript 成为 Web 开发的首选语言,最后的结果可想而知,因此到 Dart 2 发布时,已专注于改善构建客户端应用程序的体验,可以看出 Dart 定位的转变。用过 Java、Kotlin 的人,可以很快地上手 Dart。

1.2 Web 开发

Dart 是经过关键性 Web 应用程序验证的平台。它拥有为 Web 量身打造的库,如 dart:html,以及完整的基于 Dart 的 Web 框架。使用 Dart 进行 Web 开发的团队会对速度的提高感到非常激动。选择 Dart 是因为其高性能、可预测性和易学性、完善的类型系统,以及完美地支持 Web 和移动应用。

1.3 服务端开发

Dart 的服务端开发与其他的语言类似,有完整的库,可以帮助开发者快速开发服务端代码。

第 2 章

开发环境搭建

24min

接下来我们使用 Flutter 的开发环境来测试 Dart 程序。开发环境搭建还是非常烦琐的，任何一个步骤失败都会导致不能完成最终环境搭建。Flutter 支持三种环境：Windows、macOS 和 Linux。这里我们主要讲解 Windows 及 macOS 的环境搭建。

2.1 Windows 环境搭建

1. 使用镜像

首先解决网络问题。环境搭建过程中需要下载很多资源文件，当某个资源更新不成功时，就可能会导致后续报各种错误。在国内访问 Flutter 有时可能会受到限制，Flutter 官方为中国开发者搭建了临时镜像，大家可以将如下环境变量加入到用户环境变量中：

```
export PUB_HOSTED_URL = https://pub.flutter - io.cn
export FLUTTER_STORAGE_BASE_URL = https://storage.flutter - io.cn
```

注意：此镜像为临时镜像，并不能保证一直可用，读者可以参考 Using Flutter in China：https://github.com/flutter/flutter/wiki/Using-Flutter-in-China 以获得有关镜像服务器的最新动态。

2. 安装 Git

Flutter 依赖的命令行工具为 Git for Windows(Git 命令行工具)。Windows 版本的下载地址为 https://git-scm.com/download/win。

3. 下载安装 Flutter SDK

去 Flutter 官网下载其最新可用的安装包。

注意：Flutter 的渠道版本会不断更新，请以 Flutter 官网为准。Flutter 官网下载地址：https://flutter.io/docs/development/tools/sdk/archive#windows。Flutter GitHub 下载地址：https://github.com/flutter/flutter/releases。

将安装包 zip 文件解压到你想安装 Flutter SDK 的路径(如 D:\Flutter)。在 Flutter 安装目录下找到 flutter_console.bat,双击运行并启动 Flutter 命令行,接下来,你就可以在 Flutter 命令行运行 Flutter 命令了。

注意:不要将 Flutter 安装到需要一些高权限的路径,如 C:\Program Files\。

4. 添加环境变量

不管使用什么工具,如果想在系统的任意地方能够运行这个工具的命令,则需要添加工具的路径到系统环境变量 Path 里。这里路径指向 Flutter 的 bin 目录,如图 2-1 所示。同时,检查是否有名为"PUB_HOSTED_URL"和"FLUTTER_STORAGE_BASE_URL"的条目,如果没有,也需要添加它们。完成后重启 Windows 才能使更改生效。

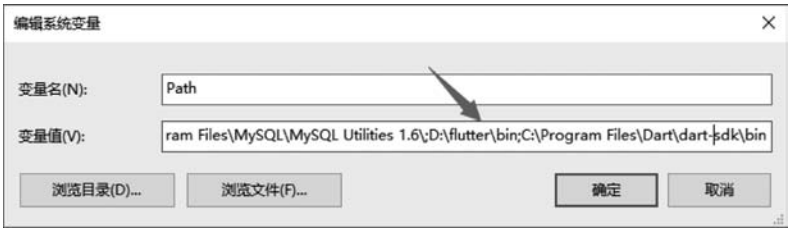

图 2-1 添加 Flutter 环境变量

5. 运行 Flutter 命令并安装各种依赖

使用 Windows 命令窗口运行以下命令,查看是否还需要安装任何其他依赖项来完成安装:

```
flutter doctor
```

该命令检查你的环境并在终端窗口中显示报告。Dart SDK 已经捆绑在 Flutter 里了,没有必要单独安装 Dart。仔细检查命令行输出以获取可能需要安装的其他软件或进一步需要执行的任务。如下显示的代码,说明 Android SDK 缺少命令行工具,需要下载并且提供了下载地址,通常这种情况只需连接网络,打开 VPN,然后重新运行"flutter doctor"命令即可。

```
[ - ] Android toolchain - develop for Android devices
    Android SDK at D:\Android\sdk
  ?Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    Try re - installing or updating your Android SDK,
     visit https://flutter.io/setup/#android - setup for detailed instructions.
```

注意:一旦你安装了任何缺失的依赖,需再次运行"flutter doctor"命令来验证你是否已经正确地设置,同时需要检查移动设备是否连接正常。

6. 编辑器设置

如果使用 Flutter 命令行工具，可以使用任何编辑器来开发 Flutter 应用程序。输入 flutter help 可查看可用的工具，但是笔者建议最好安装一款功能强大的 IDE 来进行开发，毕竟这样开发、调试、运行及打包的效率会更高。由于 Windows 环境只能开发 Flutter 的 Android 应用，所以接下来会重点介绍 Android Studio 这款 IDE。

1）安装 Android Studio

要为 Android 开发 Flutter 应用，可以使用 macOS 或 Windows 操作系统。Flutter 需要安装和配置 Android Studio，步骤如下。

步骤 1：下载并安装 Android Studio，下载地址为 https://developer.android.com/studio/index.html。

步骤 2：启动 Android Studio，然后执行“Android Studio 安装向导”。这将安装最新的 Android SDK、Android SDK 平台工具和 Android SDK 构建工具，这是用 Flutter 为 Android 开发应用时所必需的工具。

2）设置 Android 设备

要准备在 Android 设备上运行并测试 Flutter 应用，需要安装有 Android 4.1（API level 16）或更高版本的 Android 设备。

步骤 1：在设备上启用“开发人员选项”和“USB 调试”，这些选项通常在设备的“设置”界面里。

步骤 2：使用 USB 线将手机与计算机连接。如果设备出现授权提示，请授权计算机访问设备。

步骤 3：在终端中，运行 flutter devices 命令以验证 Flutter 识别所连接的 Android 设备。

步骤 4：用 flutter run 启动应用程序。

提示：默认情况下，Flutter 使用的 Android SDK 版本是基于你的 ADB 工具版本的。如果想让 Flutter 使用不同版本的 Android SDK，则必须将 ANDROID_HOME 环境变量设置为该 SDK 的安装目录。

3）设置 Android 模拟器

要准备在 Android 模拟器上运行并测试 Flutter 应用，请按照以下步骤操作。

步骤 1：启动 Android Studio→Tools→Android→AVD Manager 并选择 Create Virtual Device，打开虚拟设备面板，如图 2-2 所示。

步骤 2：选择一个设备并单击 Next 按钮，如图 2-3 所示。

步骤 3：选择一个镜像并单击 Download 按钮，然后单击 Next 按钮，如图 2-4 所示。

步骤 4：验证配置信息。填写虚拟设备名称，选择 Hardware - GLES 2.0 以启用硬件加速，单击 Finish 按钮，如图 2-5 所示。

步骤 5：在工具栏选择刚刚添加的模拟器，如图 2-6 所示。

图 2-2　打开虚拟设备面板

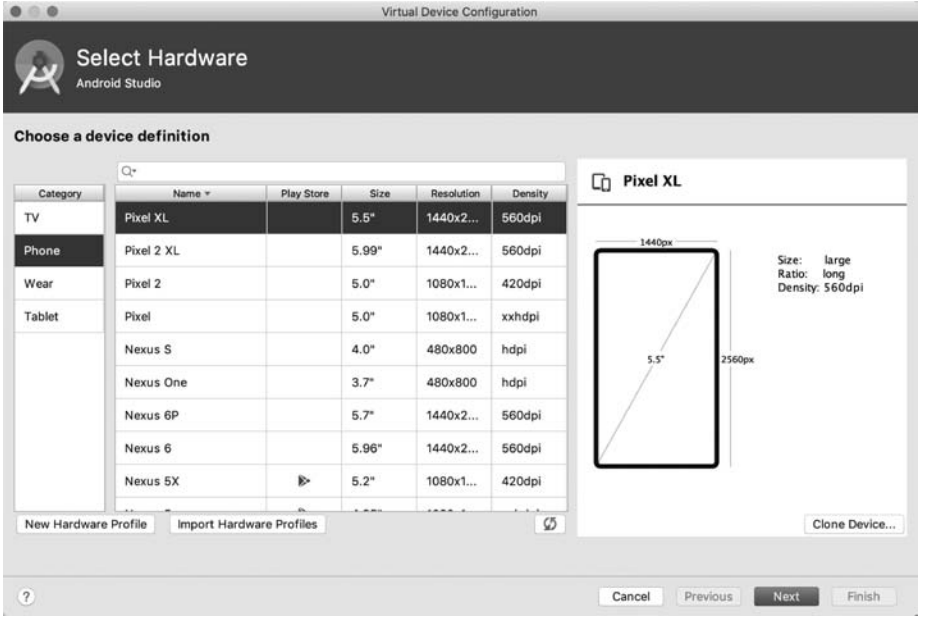

图 2-3　选择模拟硬件设备

也可以在命令行窗口运行 flutter run 启动模拟器。当能正常显示模拟器时(如图 2-7)，则表示模拟器安装正常。

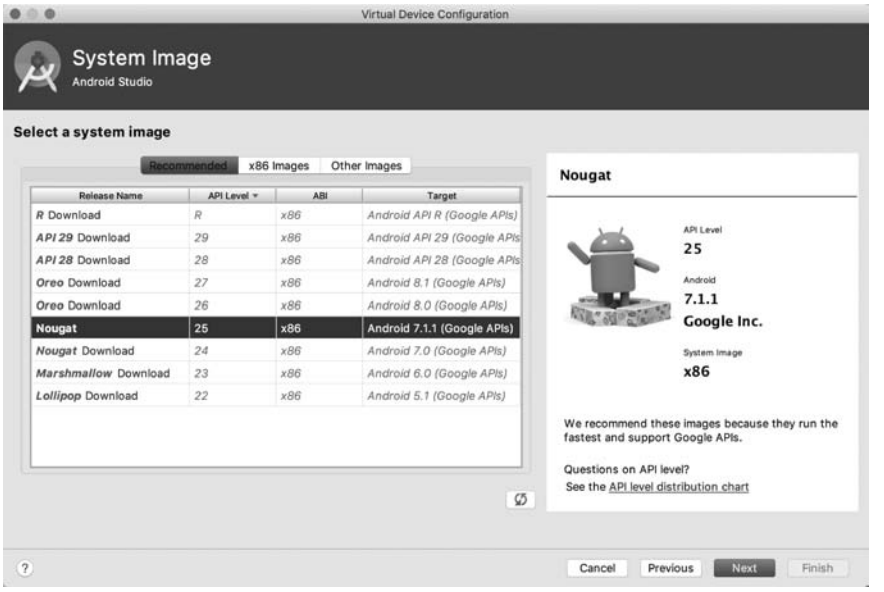

图 2-4　选择系统镜像

图 2-5　验证配置信息

提示：建议选择当前主流手机型号作为模拟器，并开启硬件加速，使用 x86 或 x86_64 镜像。详细文档请参考 https://developer.android.com/studio/run/emulator-acceleration.html。

图 2-6　在工具栏选择模拟器

图 2-7　Android 模拟器运行效果图

4）安装 Flutter 和 Dart 插件

IDE 需要安装以下两个插件：

- Flutter 插件：支持 Flutter 开发的工作流(运行、调试、热重载等)；
- Dart 插件：提供代码分析(输入代码时进行验证、代码补全等)。

打开 Android Studio 的系统设置面板，找到 Plugins，分别搜索 Flutter 和 Dart，单击安装即可，如图 2-8 所示。

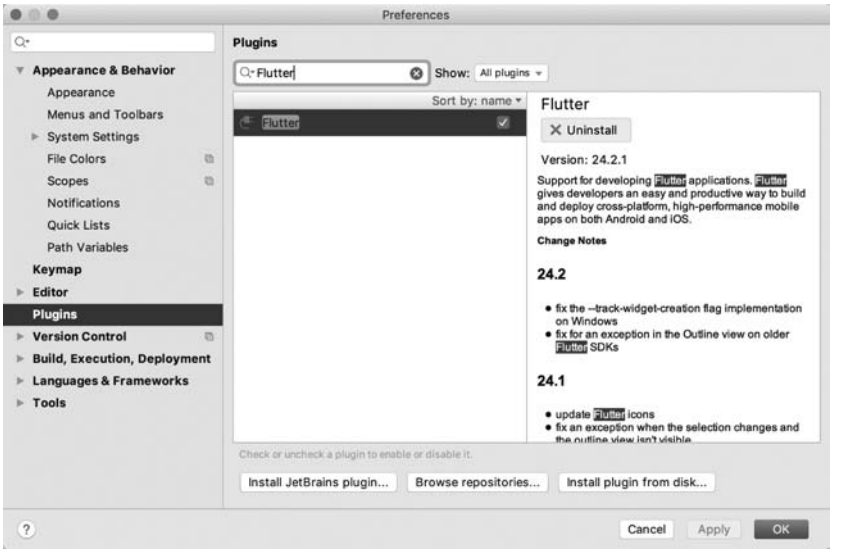

图 2-8　Android Studio 插件安装

2.2　macOS 环境搭建

首先解决网络问题，参见 2.1 节“Windows 环境搭建”。

1. 命令行工具

Flutter 依赖的命令行工具有 bash、mkdir、rm、git、curl、unzip 和 which。

2. 下载安装 Flutter SDK

请按以下步骤下载并安装 Flutter SDK。

步骤 1：去 Flutter 官网下载其最新可用的安装包。

注意：Flutter 的渠道版本会不断更新，请以 Flutter 官网为准。另外，在中国大陆地区，要想获取安装包列表或下载安装包有可能会遇到困难，读者也可以去 Flutter GitHub 项目下去下载安装 Release 包。

Flutter 官网下载地址：https://flutter.io/docs/development/tools/sdk/archive#macOS。

Flutter GitHub 下载地址：https://github.com/flutter/flutter/releases。

步骤 2：解压安装包到想安装的目录，如：

```
cd /Users/ksj/Desktop/flutter/
unzip /Users/ksj/Desktop/flutter/v0.11.9.zip.zip
```

步骤 3：添加 Flutter 相关工具到 PATH 中：

```
export PATH = 'pwd'/flutter/bin: $ PATH
```

3. 运行 Flutter 命令并安装各种依赖

运行以下命令查看是否需要安装其他依赖项：

```
flutter doctor
```

该命令检查你的环境并在终端窗口中显示报告。Dart SDK 已经捆绑在 Flutter 里了，没有必要单独安装 Dart。仔细检查命令行输出以获取可能需要安装的其他软件或进一步需要执行的任务(以粗体显示)。如下代码提示表示 Android SDK 缺少命令行工具，需要下载并且提供了下载地址，通常这种情况只需要把网络连好，VPN 开好，然后重新运行 flutter doctor 命令。

```
[ - ] Android toolchain - develop for Android devices
    Android SDK at /Users/obiwan/Library/Android/sdk
  ?Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    Try re - installing or updating your Android SDK,
visit https://flutter.io/setup/#android - setup for detailed instructions.
```

注意：当安装了所缺失的依赖后，需再次运行 flutter doctor 命令来验证是否已经正确地设置，同时需要检查移动设备是否连接正常。

4. 添加环境变量

使用 vim 命令打开 ～ /. bash_profile 文件，添加如下内容：

```
export ANDROID_HOME = ～/Library/Android/sdk                    //android sdk 目录
export PATH = $ PATH: $ ANDROID_HOME/tools: $ ANDROID_HOME/platform - tools
export PUB_HOSTED_URL = https://pub.flutter - io.cn            //国内用户需要设置
export FLUTTER_STORAGE_BASE_URL = https://storage.flutter - io.cn //国内用户需要设置
export PATH = /Users/ksj/Desktop/flutter/flutter/bin: $ PATH //直接指定 flutter 的 bin 地址
```

注意：请将 PATH＝/Users/ksj/Desktop/flutter/flutter/bin 更改为你的路径。

完整的环境变量设置如图 2-9 所示。

```
export ANDROID_HOME=~/Library/Android/sdk
export PATH=$PATH:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools
export PUB_HOSTED_URL=https://pub.flutter-io.cn
export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
export PATH=/Users/ksj/Desktop/flutter/flutter/bin:$PATH
export PATH=$PATH:$HOME/.pub-cache/bin

# Setting PATH for Python 3.7
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.7/bin:${PATH}"
export PATH

export GOPATH=/Users/ksj/Desktop/yu/golang
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOBIN


export M2_HOME=/Users/ksj/Desktop/software/maven/
export PATH=$PATH:$M2_HOME/bin

export GO111MODULE=on
#export GOPROXY=https://goproxy.io
export GOPROXY=https://mirrors.aliyun.com/goproxy
```

图 2-9　macOS 环境变量设置

设置好环境变量以后，请务必运行 source $ HOME/. bash_profile 刷新当前终端窗口，以使刚刚配置的内容生效。

5. 编辑器设置

如果使用 Flutter 命令行工具，可以使用任何编辑器来开发 Flutter 应用程序。输入 flutter help 可查看可用的工具，但是笔者建议最好安装一款功能强大的 IDE 来进行开发，毕竟这样开发、调试、运行和打包的效率会更高。由于 macOS 环境既能开发 Android 应用也能开发 iOS 应用，所以 Android 设置请参考 2. 1 节“Windows 环境搭建”中的“安装

Android Studio”，接下来仅介绍 Xcode 的使用方法。

1）安装 Xcode

安装最新 Xcode。通过链接下载：https://developer.apple.com/xcode/，或通过苹果应用商店下载：https://itunes.apple.com/us/app/xcode/id497799835。

2）设置 iOS 模拟器

要在 iOS 模拟器上运行并测试你的 Flutter 应用，需要打开一个模拟器。在 macOS 的终端输入以下命令：

```
open -a Simulator
```

可以找到并打开默认模拟器。如果想切换模拟器，可以打开 Hardware 并在其下的 Device 菜单选择某一个模拟器，如图 2-10 所示。

打开后的模拟器如图 2-11 所示。

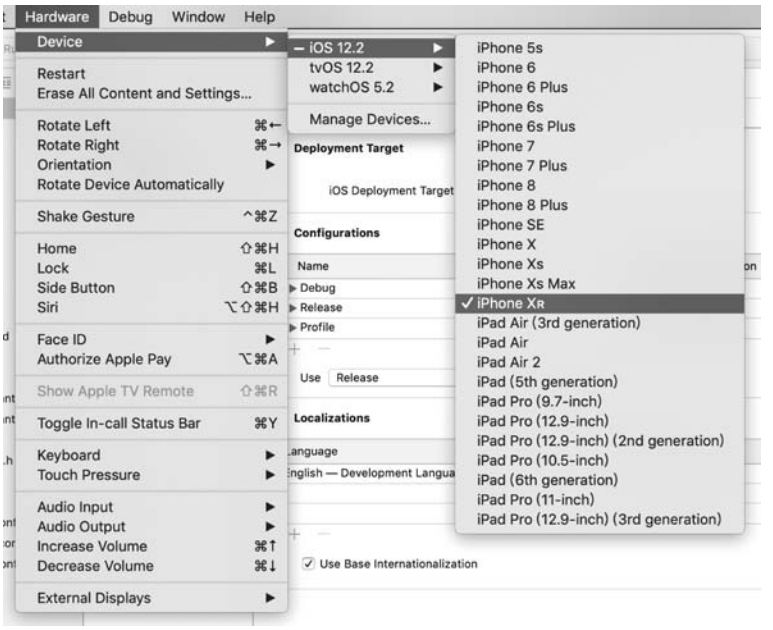

图 2-10　选择 iOS 模拟器

图 2-11　iOS 模拟器效果图

接下来，在终端运行 flutter run 命令或者打开 Xcode，如图 2-12 所示，选择好模拟器。单击 Runner 按钮即可启动你的应用。

3）安装到 iOS 设备

要在苹果真机上测试 Flutter 应用，需要一个苹果开发者账户，并且还需要在 Xcode 中进行设置。

（1）安装 Homebrew 工具。Homebrew 是一款 macOS 平台下的软件包管理工具，拥有

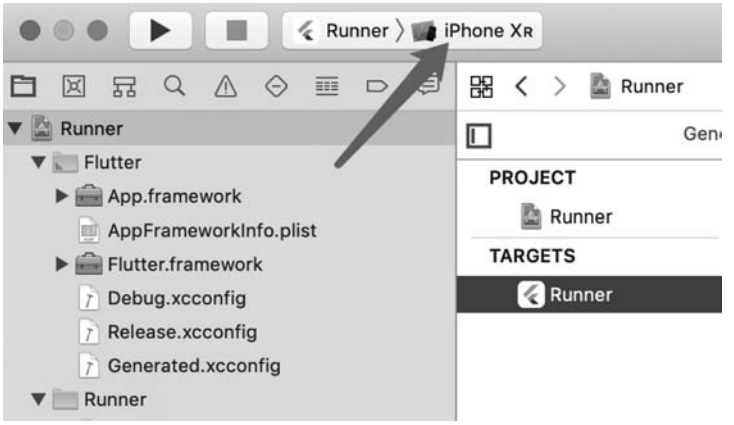

图 2-12　Xcode 启动应用

安装、卸载、更新、查看和搜索等很多实用的功能。下载地址为 https://brew.sh。

(2) 打开终端并运行一些命令,安装用于将 Flutter 应用安装到 iOS 设备的工具,命令如下所示:

```
brew update
brew install --HEAD libimobiledevice
brew install ideviceinstaller ios-deploy cocoapods
pod setup
```

提示:如果这些命令中有任何一个失败并出现错误,请运行 brew doctor 并按照说明解决问题。

接下来需要 Xcode 签名。Xcode 签名设置有以下几个步骤。

步骤 1:在你的 Flutter 项目目录中双击 ios/Runner.xcworkspace 打开默认的 Xcode 工程。

步骤 2:在 Xcode 中,选择导航面板左侧中的 Runner 项目。

步骤 3:在 Runner TARGETS 设置页面中,确保在 General→Signing→Team(常规→签名→ 团队)下选择了你的开发团队,如图 2-13 所示。当你选择一个团队时,Xcode 会创建并下载开发证书,为你的设备注册你的账户,并创建和下载配置文件。

步骤 4:要开始你的第一个 iOS 开发项目,可能需要使用你的 Apple ID 登录 Xcode。任何 Apple ID 都支持开发和测试。需要注册 Apple 开发者计划才能将你的应用分发到 App Store,具体方法请查看 https://developer.apple.com/support/compare-memberships/这篇文章。Apple ID 登录界面如图 2-14 所示。

步骤 5:当你第一次添加真机设备进行 iOS 开发时,需要同时信任你的计算机和该设备上的开发证书。单击 Trust 按钮即可,如图 2-15 所示。

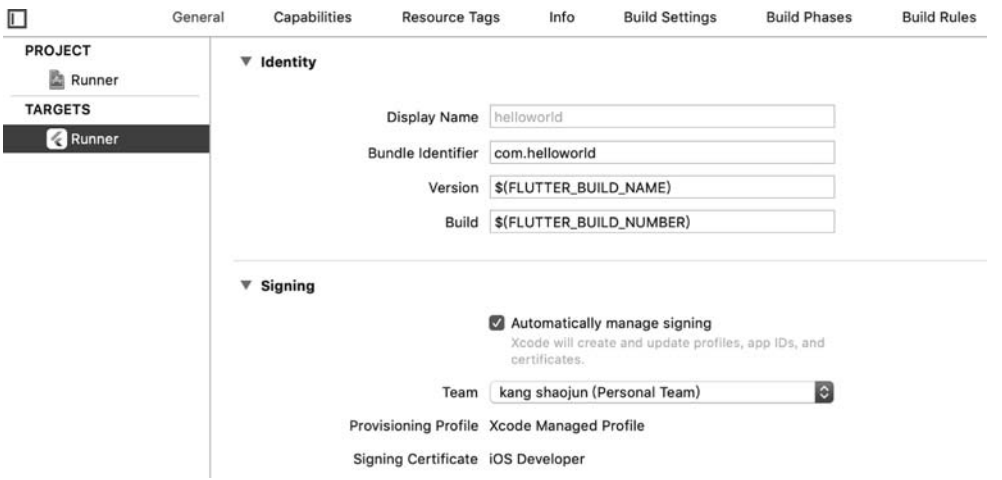

图 2-13　设置开发团队

图 2-14　使用 Apple ID

图 2-15　信任此计算机图示

步骤 6：如果 Xcode 中的自动签名失败，请查看项目的 Bundle Identifier 值是否唯一。这个 ID 即为应用的唯一 ID，建议使用域名反过来写，如图 2-16 所示。

步骤 7：使用 flutter run 命令运行应用程序。

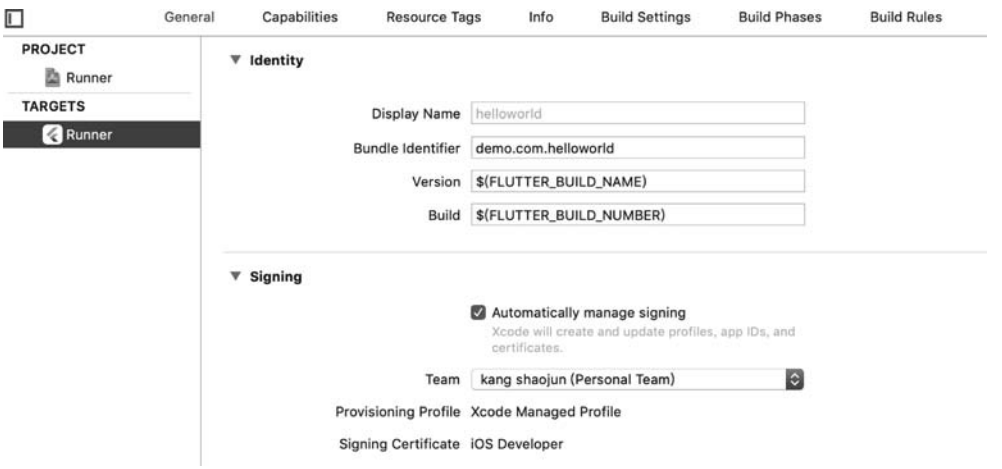

图 2-16　验证 Bundle Identifier 值

第 3 章

第一个 Dart 程序

万事开头难，我们用输出“Hello World”来看一个最简单的 Flutter 工程，具体步骤如下。

步骤 1：新建一个 Flutter 工程，选择 Flutter Application，如图 3-1 所示。

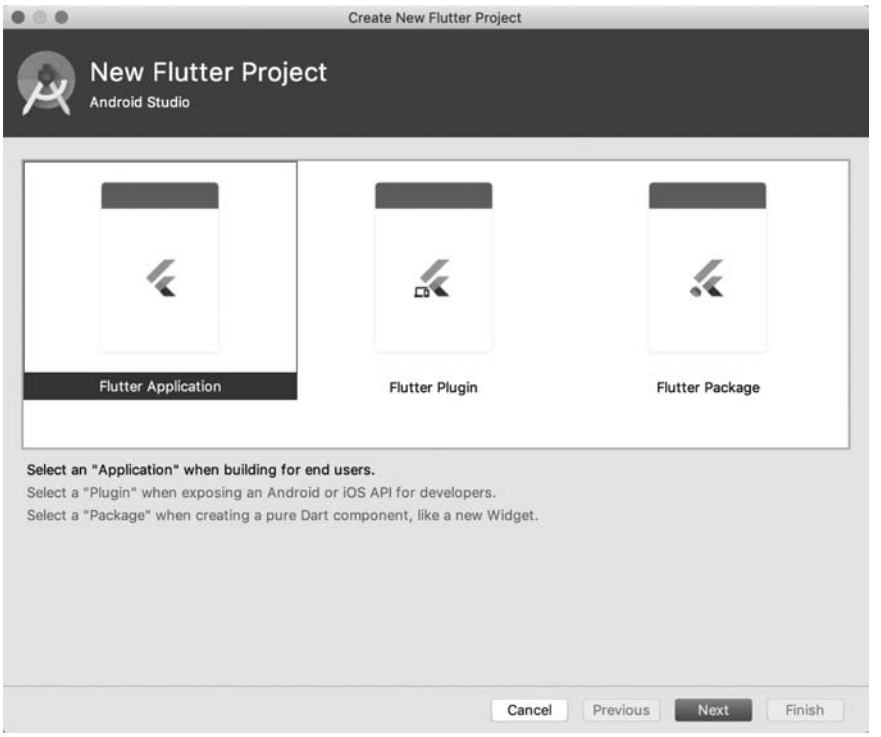

图 3-1　新建工程

步骤 2：单击 Next 按钮，打开应用配置界面，其中在 Project name 中填写 helloworld，Flutter SDK path 使用默认值，IDE 会根据 SDK 安装路径自动填写，Project location 填写为工程放置的目录，在 Description 中填写项目描述，任意字符即可，如图 3-2 所示。

步骤 3：单击“Next”按钮，打开包设置界面，在 Company domain 中填写域名，注意域名

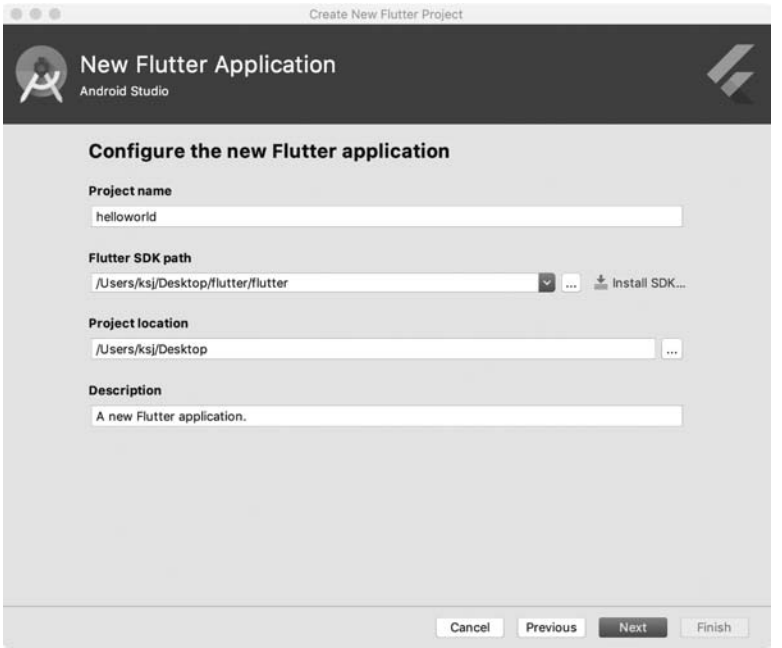

图 3-2　配置 Flutter 工程

要反过来写,这样可以保证全球唯一,Platform channel language 下面的两个选项不需要勾选,如图 3-3 所示。

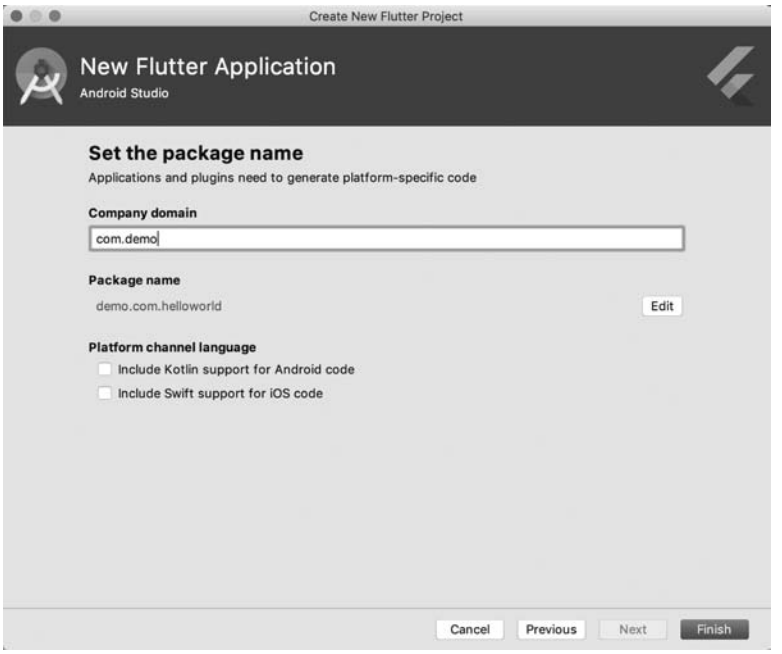

图 3-3　设置包名界面

步骤 4：单击 Finish 按钮开始创建第一个工程，等待几分钟，会创建如图 3-4 所示工程。

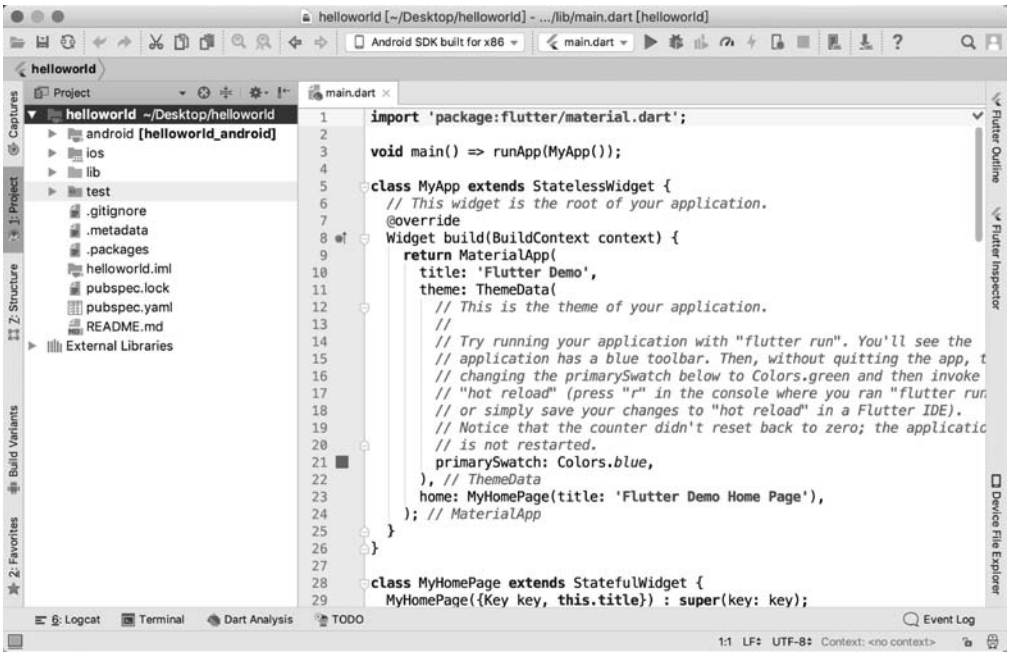

图 3-4　示例工程主界面

步骤 5：工程建好后，先运行一下，看一看根据官方推荐方案所创建的示例的运行效果，单击 Open iOS Simulator 命令打开 iOS 模拟器，具体操作如图 3-5 所示。

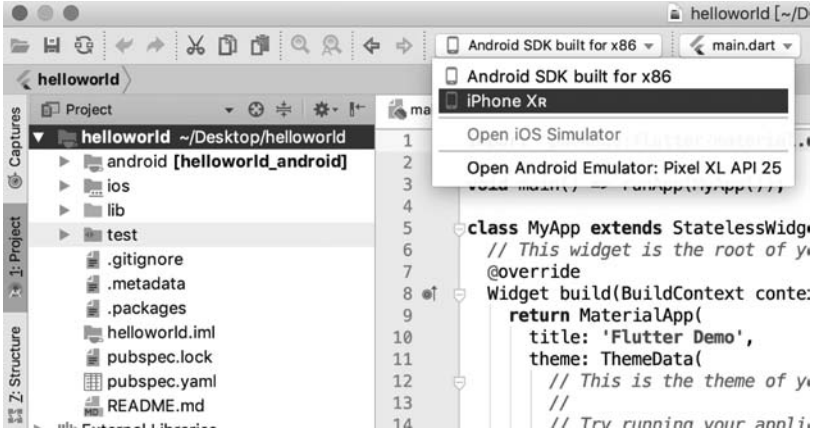

图 3-5　打开模拟器菜单示意图

步骤 6：等待几秒后会打开模拟器，如图 3-6 所示。

步骤 7：打开工程目录下的 lib/main.dart 文件，删除所有代码并替换成如下代码：

图 3-6　模拟器启动完成图

```
//main.dart 文件
//程序执行入口函数
main() {
  //定义并初始化变量
  String msg = 'Hello World';
  //调用方法
  sayHello(msg);
}
//方法
sayHello(String msg) {
  //在控制台打印内容
  print('$msg');
}
```

步骤 8：单击 debug(调试)按钮，启动 Hello World 程序控制台输出“Hello World”，这样，第一个 Dart 程序就运行出来了，输出内容如下：

```
Performing hot restart...
Syncing files to device iPhone XR...
Restarted application in 52ms.
flutter: Hello World
```