第 2 章

开发环境

亲爱的读者，本章将重点介绍 Flutter 应用开发环境的配置。

2.1 下载开发工具包

在 Flutter 的官方网站（flutter.dev）或 Flutter 文社区网站，可以下载 Flutter 的 SDK（Software Development Kit，软件开发工具包）。下载文件通常就是一个“.zip”格式的压缩包，把解压后的 flutter 目录放在系统的某个目录下，再配置好系统的环境变量目录，即可在任何地方执行 flutter 命令。

下载开发工具包时要选择适合个人计算机操作系统的版本，如 Windows、macOS 或 Linux 版本的 Flutter。下载得到的压缩包文件名通常是 flutter_macos_3.7.6-stable.zip 这样的形式，其中 macos 指的是操作系统，3.7.6 是 Flutter SDK 的版本号，stable 表示这是一个稳定版本的 Flutter。除了稳定版，还有测试版（beta）与开发版（dev）。

2.1.1 任务：macOS 系统下安装 Flutter

下面介绍如何在 macOS 系统下配置使用 Flutter。

1．下载 Flutter

在 Flutter 官方网站（flutter.dev）或 Flutter 中文社区网站上下载最新的适合在 macOS 平台使用的 Flutter SDK。注意在下载页面会提供两个适用于 macOS 平台的 Flutter SDK，如果您的计算机用的是 M1/M2 芯片，可以下载带 arm64 字样的 Flutter SDK，如 flutter_macos_arm64_3.7.6-stable.zip。

2．存放 Flutter

下载的文件是一个压缩包，解压以后把它放在/Applications 系统目录下（该目录下有很多应用程序）。在终端，执行命令：

```
cd /Applications
unzip ~/Downloads/flutter_macos_*.zip
```

进入系统根目录的 Applications 目录，用系统自带的 unzip 软件解压主目录下 Downloads 目录里的 Flutter SDK 压缩包。注意，解压时可以用通配符*省略部分文件名，当然也可以具体设置要解压的压缩包文件名。另外，如果是用 Safari 浏览器下载的 Flutter SDK，下载完成后很可能会自动对压缩包进行解压。

不管怎么样，我们要做的就是下载 Flutter SDK，把解压之后得到的 flutter 目录放到/Applications 这个目录里面。

3．测试执行 flutter 命令

在终端，执行如下命令：

```
/Applications/flutter/bin/flutter
```

上面这行命令可执行/Applications/flutter/bin 目录下的 flutter 命令行工具，显示它的帮助信息。读者也可以试一下能否在不给出完整路径的情况下，直接执行 flutter 命令，通常会提示找不到 flutter 命令。要想在任意地方都能直接执行 flutter 命令，需要配置系统的环境变量目录。

4．配置系统环境变量目录

把 flutter 命令所在的目录路径设置成系统环境变量目录，这样就可以随时随地直接使用 flutter 命令。

用编辑器编辑用户主目录下的终端配置文件。例如，可以用 VSCode 提供的 code 命令打开要编辑的文件：

```
code ~/.zprofile
```

也可以用 vi 命令编辑~/.zprofile 文件：

```
vi ~/.zprofile
```

在这个配置文件里，添加如下配置信息：

```
export PATH="$PATH:/Applications/flutter/bin"
```

上面这行配置信息的含义是把/Applications/flutter/bin 目录作为系统环境变量目录。在终端执行命令时，系统会在环境变量目录里搜寻要执行的命令，找到匹配的命令就会执行它。

保存文件并重启终端，然后再次直接执行 flutter 命令，如果终端显示这个命令的帮助信息，就表明已在 macOS 上安装配置好了 Flutter。

2.1.2 任务：Windows 系统下安装 Flutter

下面介绍如何在 Windows 系统下配置使用 Flutter。

1．下载 Flutter

在 Flutter 官方网站（flutter.dev）或 Flutter 中文社区网站上下载最新的适合在 Windows 平台使用的 Flutter SDK。

2．存放 Flutter

解压下载的 Flutter SDK 压缩包，会得到一个 flutter 目录，把这个目录放在系统某个目录下，如 C 盘的根目录下面（C:\）。

3．测试执行 flutter 命令

在终端（Cmder）执行如下命令：

```
/c/flutter/bin/flutter
```

上面这行命令可执行/c/flutter/bin 目录下的 flutter 命令行工具，终端会显示它的帮助信息。同样，读者可以试一下能否不给出完整路径，直接执行 flutter 命令。很遗憾，终端一样会提示找不到 flutter 命令。要想在任意地方都能直接执行 flutter 命令，需要配置系统的环境变量目录。

4．配置系统环境变量目录

把 flutter 命令所在的目录路径设置成系统的环境变量目录，这样就可以随时随地直接使用 flutter 命令。

在 Windows 系统下，我们使用 Cmder 作为命令行界面，之前安装时把它放在了 C:\Program Files 目录下，打开该目录下的 config/user_profile.sh 配置文件，然后添加如下配置信息：

```
export PATH=/c/flutter/bin:${PATH}
```

上面这行配置信息的含义是把/c/flutter/bin 目录作为系统的环境变量目录，在终端执行命令时，系统会在环境变量目录里寻找要执行的命令。

保存文件并重启终端（Cmder），然后再次直接执行 flutter 命令，如果终端显示命令的帮助信息，则表明已在 Windows 系统下安装配置好了 Flutter。

2.1.3　任务：配置使用国内镜像

在下载升级 Flutter SDK，安装管理项目软件包的时候，我们默认会从 Flutter 官方网址下载，但有时候网速会不稳定，所以也可以使用国内提供的镜像下载链接，这样下载速度会更快一些。

要想使用国内镜像，需要在系统里设置两个环境变量。macOS 用户需要编辑~/.zprofile 终端配置文件，Windows 用户需要编辑 cmder/config/user_profile.sh 终端配置文件。

在配置文件里添加如下两行内容：

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

如果上述镜像无法启用，可以换用上海交通大学提供的如下镜像地址：

```
export PUB_HOSTED_URL=https://mirrors.sjtug.sjtu.edu.cn
export FLUTTER_STORAGE_BASE_URL=https://dart-pub.mirrors.sjtug.sjtu.edu.cn
```

2.2　准备 iOS 与 macOS 应用开发环境

使用 Flutter 可以开发能运行在 iOS 与 macOS 平台上的应用，即可以在 iPhone 与 Mac

计算机上使用的应用。我们需要在一台 Mac 计算机上配置 iOS 与 macOS 应用的开发环境。如果读者使用的计算机是 Windows 操作系统，可以跳过这一步，直接配置 Android 与 Web 平台应用的开发环境。

2.2.1 任务：安装 Rosetta

如果您是在 Apple Silicon 版本的 Mac 上运行 Flutter，也就是如果您的 Mac 计算机用的是 M1/M2 芯片，则需要用到 Rosetta 翻译环境，启用这个环境可以在终端执行如下命令：

```
sudo softwareupdate --install-rosetta --agree-to-license
```

2.2.2 任务：安装 Homebrew

Homebrew 是 macOS 系统的一种软件包管理工具，常用来安装命令行工具。换句话说，Homebrew 不是必须的工具，但却是一个很好的帮手。

在 macOS 系统里安装 Homebrew，只需要执行 Homebrew 官方网站提供的一行脚本即可。在 brew.sh 网站可以找到这行要执行的脚本，复制这行脚本并打开终端执行一下。Homebrew 安装完成以后，就可以在终端使用 brew 命令管理系统的软件包了。

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/
install/HEAD/install.sh)"
```

2.2.3 任务：安装与准备 Xcode

Xcode 是开发 iOS 与 macOS 应用必须的工具。

1．安装 Xcode

打开 macOS 系统里的应用商店（App Store），搜索并安装 Xcode。这个软件体积较大，安装需要一段时间。安装完成后，打开 Xcode 软件，会提示安装一些额外的组件，单击 Install 按钮，确认安装这些额外的组件即可。

2．准备 Xcode

安装 Xcode 以后，在终端执行如下命令：

```
flutter doctor
```

上面这行命令可以检查 Flutter 的开发环境。先观察命令执行结果里出现的 Xcode，如果有下面这样的提示，说明已经准备好了 Xcode。

```
[✓] Xcode - develop for iOS and macOS
```

有时会给出一些安装问题提示，如 Xcode 安装的不完整、Cocoapods 版本太低等。仔细阅读问题说明，然后根据这些提示解决存在的问题。

如果提示 Xcode installation is incomplete，即 Xcode 安装的不完整，可执行下面两个命令：

```
sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
sudo xcodebuild -runFirstLaunch
```

如果提示 Cocoapods 版本太低，它是 Swift 语言的包管理工具。要使用新版本的 Cocoapods，可以用 Homebrew 去安装一个，执行：

```
brew install cocoapods
```

命令运行完成以后，再执行：

```
brew link --overwrite cocoapods
```

确定系统使用的 Cocoapods 版本，可以执行：

```
pod --version
```

升级了新版本的 Cocoapods 以后，再次执行 flutter doctor 命令，观察 Xcode 是否已经准备好了，如果没问题，会在 Xcode 前面出现[✓]。准备好 Xcode 后，就可以用 Flutter 开发 iOS 或 macOS 平台的应用了。

2.3　准备 Android 平台应用开发环境

用 Flutter 可以开发适用于 Android 平台的 App，即可以在安卓手机上使用的应用程序。在 Windows 与 macOS 系统上都可以基于 Flutter 开发 Android 应用，开发时需要用到 Android Studio 提供的一些功能。

下面介绍如何下载和安装 Android Studio。

1．安装 Android Studio

访问 Android 开发者网站（https://developer.android.google.cn/studio），下载适用于个人操作系统的 Android Studio，把它安装在自己的计算机上以后，再打开它。

2．配置 Android Studio

第一次打开 Android Studio 时需要做一些配置（可以后续再修改）。

下载 Android SDK 时，国内用户可能会遇到网络问题，提示 Unable to access Android SDK add-on list，即无法访问 Android SDK 扩展列表（见图 2.1），这个问题可以通过配置代理来解决。

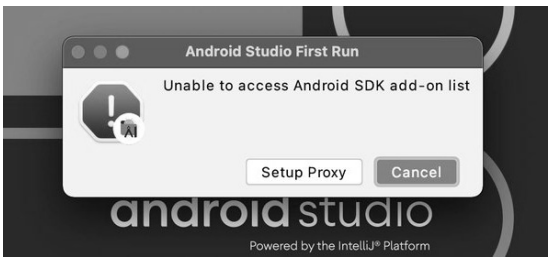

图 2.1　遇到网络连接问题会提示配置代理（Proxy）

单击 Setup Proxy 按钮，打开代理配置界面，如图 2.2 所示。选中 Manual proxy configuration（手动配置代理）单选按钮，然后输入代理服务的 Host name（主机名）和 Port number（端口号）。如果无法直接通过网络访问 Android SDK 列表，就需要通过代理服务来访问这个列表。注意，这个代理服务需要个人准备好。

配置好代理服务后，单击 OK 按钮，如果一切正常，可以访问扩展列表，此时 Android Studio 会启动配置，一路按 Enter 键即可完成对应配置。配置完成后，Android Studio 会下载 SDK 和相关工具，如图 2.3 所示。

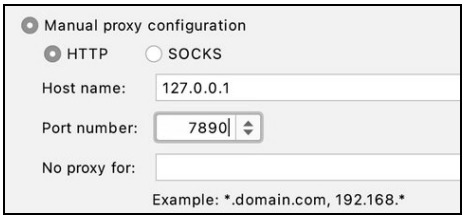

图 2.2　手工配置 Android Studio 代理

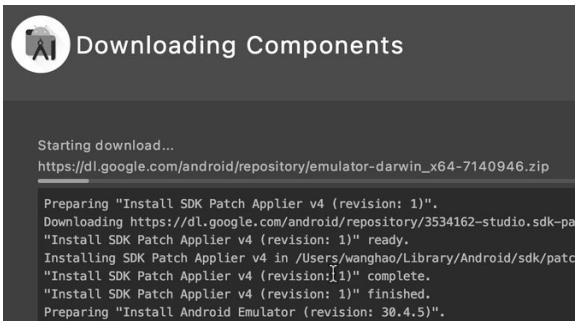

图 2.3　正在下载组件

3．确认已准备好 Android 工具链

安装配置好 Android Studio 后，基本上就准备好了使用 Flutter 开发 Android 平台应用需要的工具。在终端执行如下命令：

```
flutter doctor
```

观察 Android toolchain 出现的问题，很可能会提示 Android licenses not accepted，意思是有些协议条款还需确认一下。执行命令：

```
flutter doctor --android-licenses
```

执行上述命令，会提示很多次确认同意条款，输入 y，然后按 Enter 健表示确认。完成以后，再次执行 flutter doctor 命令，准备好 Android 工具链以后，会在 Android toolchain 前面出现[✓]，像下面这样。

```
[✓] Android toolchain - develop for Android devices ...
```

2.4　准备设备模拟器

用户可以在真实的 iOS 与 Android 设备上运行调试 Flutter 应用，也可以在模拟器上运行调试 Flutter 应用。安装了 Xcode 后，可以直接使用 iOS 设备的模拟器。通过 Android Studio 可以创建一些 Android 模拟器。

在终端查看可用的设备模拟器，可以执行：

```
flutter emulators
```

正常的话，列出的内容像下面这样：

```
2 available emulators:
apple_ios_simulator • iOS Simulator    • Apple   • ios
Pixel_XL_API_28     • Pixel XL API 28 • Google • android
```

上面的信息提示系统里有两台模拟器，一个是 iOS 平台模拟器，ID 是 apple_ios_simulator；

另一个是 Android 平台模拟器，ID 是 Pixel_XL_API_28。开发 Flutter 应用时，可以在这些模拟器上调试应用程序。

配置 Android 模拟器的操作如下。

打开 Android Studio，在启动界面上单击右下角的 Configure 按钮，在弹出的菜单中选择 AVD Manager 选项（见图 2.4），即可管理 Android 模拟器。

注意：AVD 是 Android Virtual Device（安卓虚拟设备）的缩写，也就是我们说的模拟器。

通过 AVD Manager 选项可以添加或编辑模拟器。新建一个 Android 模拟器，名字设置为 flutter，如图 2.5 所示。创建了 Android 模拟器后，在终端执行 flutter emulators 命令，会显示刚才新建的 Android 模拟器。

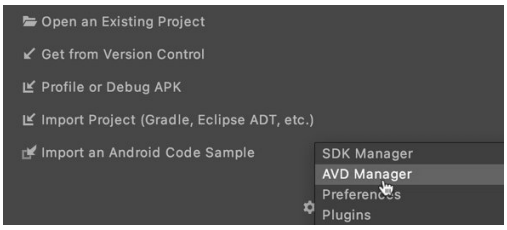

图 2.4　安卓虚拟设备管理器

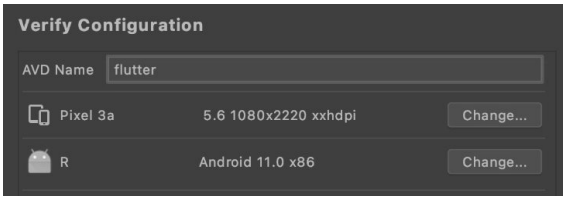

图 2.5　创建安卓虚拟设备

2.5　准备 Web 应用开发环境

用 Flutter 可以开发 Web 应用，也就是可以在浏览器上运行的应用。要开发 Web 应用，需要先在计算机上安装 Chrome 浏览器。

Chrome 是一款备受开发者青睐的浏览器，可在百度搜索 chrome 关键字，或通过网址 https://www.google.cn/chrome/下载并安装 Chrome 浏览器。安装好以后，执行 flutter doctor 命令，正常的话会在 Chrome 前面显示[✓]，此时就可以基于 Flutter 开发 Web 应用了。

```
[✓] Chrome - develop for the web
```

2.6　准备代码编辑器 VSCode

VSCode 是本书使用的代码编辑器。要使用这款编辑器开发 Flutter 应用，需要给编辑器安装一个 Flutter 插件。

打开 VSCode 编辑器，再打开编辑器的扩展管理，然后搜索并安装 Flutter 插件（见图 2.6）。安装好以后，就可以使用 VSCode

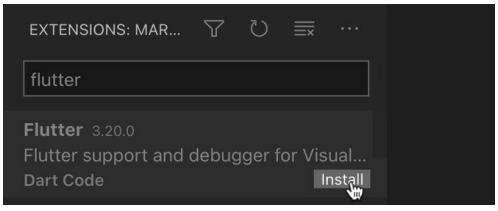

图 2.6　在 VSCode 编辑器安装 Flutter 插件

编辑器编写 Flutter 应用代码了。

2.7 创建 Flutter 项目

给 VSCode 编辑器装好 Flutter 插件后，打开编辑器的命令面板，搜索并执行 Flutter: New Application Project 命令，可以创建一个全新的 Flutter 项目。同样，也可以在命令行界面使用 flutter create 命令创建一个 Flutter 项目。

2.7.1 任务：创建并运行 Flutter 项目

1．创建一个 Flutter 项目

在终端进入想要保存项目的目录，然后用 flutter create 创建一个 Flutter 项目。命令如下：

```
cd ~/desktop
flutter create xb2_flutter
```

执行上面两行命令后，将在桌面上创建一个 Flutter 项目，后文中再提及“在项目所在目录的下面”，指的就是桌面上的 xb2_flutter 目录（即~/desktop/xb2_flutter）。

2．用编辑器打开项目

进入新创建的 Flutter 项目所在目录，然后用 VSCode 编辑器打开这个项目：

```
cd ~/desktop/xb2_flutter
code ./
```

3．运行 Flutter 应用

打开项目的 src/main.dart 文件，这是应用的入口文件。单击编辑器底部右下角出现的设备名称，如 Chrome（web-Javascript），会打开一个设备列表（见图 2.7），这里选择 Chrome。

打开 VSCode 编辑器的调试功能，按 Shift+Command+D 或 Shift+Ctrl+D 快捷键，单击 Run and Debug 按钮（见图 2.8），成功后会在 Chrome 浏览器上打开 Flutter 应用（见图 2.9）。

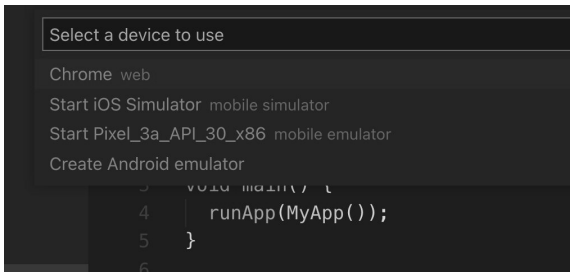

图 2.7　在 VSCode 编辑器选择要使用的设备

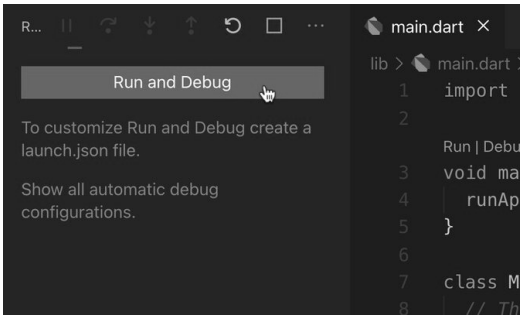

图 2.8　在 VSCode 编辑器运行调试

4．测试编辑应用

回到编辑器，打开 src/main.dart 文件，修改 MyHomePage 小部件 title 参数值为 ninghao.net。

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      ...
      home: MyHomePage(title: 'ninghao.net'),
    );
  }
}
```

保存文件，观察 Chrome 浏览器运行的应用界面，会发现界面顶部显示的标题已变成 ninghao.net，如图 2.10 所示。

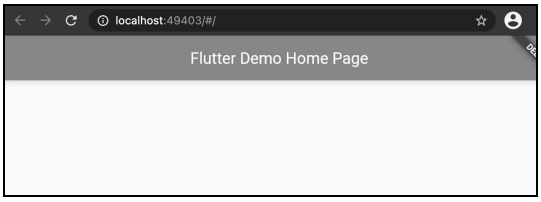

图 2.9　在 Chrome 浏览器中调试 Flutter 应用

图 2.10　修改了标题以后的结果

5．在模拟器上运行调试应用

停止编辑器的调试，选择一个模拟器设备，然后重新执行调试功能，此时将在模拟器上运行正在开发的 Flutter 应用（见图 2.11）。编辑项目文件，保存文件以后，可以实时地看到界面的变化。

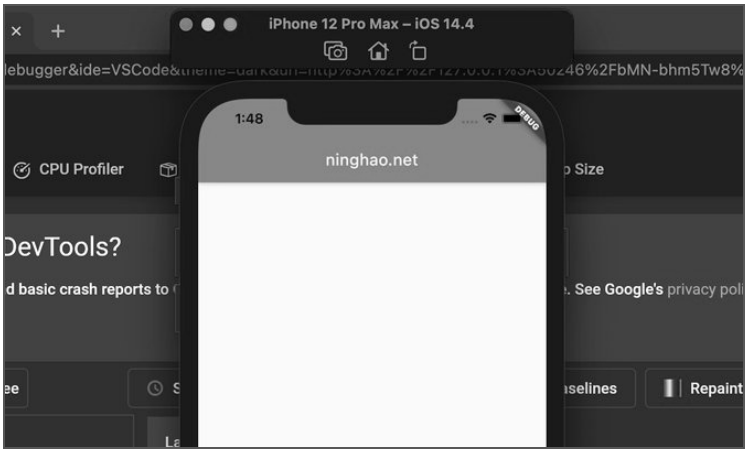

图 2.11　在 iOS 模拟器上调试 Flutter 应用

如果需要重启应用调试，可以在打开编辑器的调试（单击 Run and Debug 按钮）功能后，单击“重启”按钮，或者按 Shift+Command+F5 快捷键（在 macOS 下）。

2.7.2 任务：清理项目与源代码管理

可以使用 Git 对项目源代码进行管理，这个不是必须的，但推荐大家学习一下。

1．清理 main.dart 文件

main.dart 是应用的入口文件，打开这个文件，然后把文件里的所有内容全部删除。

2．删除 test 目录

在 test 目录里有一些测试文件，可以暂时把这个 test 目录全部删除。

3．创建仓库并做一次初始提交

在终端，进入项目所在目录下：

```
cd ~/desktop/xb2_flutter
```

然后初始化项目仓库，执行：

```
git init
```

做一次初始提交：

```
git add .
git commit -m 'init'
```

4．添加远程仓库

在 Github 或 coding.net 中给项目创建一个远程仓库，然后在项目本地仓库里添加这个远程仓库。例如，在 Github 中给项目创建了一个远程仓库，地址是 git@github.com:ninghao/xb2_flutter.git，在终端项目所在目录的下面，执行：

```
git remote add origin git@github.com:ninghao/xb2_flutter.git
```

执行上面的命令，可以给项目添加一个名为 origin 的远程仓库，这个仓库就是在 Github 上面创建的。给项目添加了远程仓库后，就可以把项目的本地仓库推送到远程仓库里了。

2.8 问题与思考

问题 1：完成任务后该如何提交

跟随本书开发应用的时候，每完成一个任务，如果任务里修改了项目文件，就需要做一次提交（commit），保存一下项目当前的状态。提交的时候可以编写一条日志，说明一下本次提交对项目所做的修改部分，建议使用任务标题作为提交日志的内容。读者可以在命令行下完成提交，也可以使用 VSCode 编辑器的源代码管理（source control）功能。

（1）在命令行下提交任务。

在终端执行命令：

```
git status
git add .
git commit -m '描述一下这次提交'
```

执行 git status 命令，可以查看当前项目的状态，如项目里哪些文件发生了变化，新建或删除了哪些文件等。然后选择本次提交包含的修改，执行“git add .”命令，添加当前发生的所有修改。确定提交后，执行 git commit 命令，用-m 选项设置一条提交日志。

（2）使用编辑器源代码管理功能。

VSCode 编辑器自带源代码管理功能，完成任务以后，打开编辑器的源代码管理界面，如图 2.12 所示。

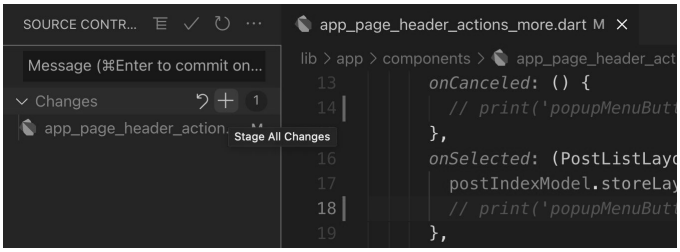

图 2.12　VSCode 编辑器的源代码管理功能

Changes 列表会列出项目当前有变化的地方（文件），单击右侧的+按钮（Stage All Changes），可以将当前发生的所有修改包含在这次要做的提交里，如图 2.13 所示。

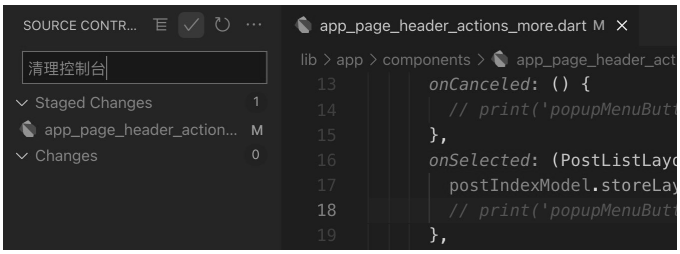

图 2.13　选中要包含在提交里的修改并设置提交日志

添加修改后，在 Staged Changes 下可以看到本次添加的修改，即要包含在本次提交里的修改。在上面输入一条日志，描述本次提交要做的事情，然后单击 √ 按钮，确定提交。

问题 2．如何使用已完成的项目

读者可以参考笔者已经完成的 Flutter 项目，里面包含了所有要在本书中编写的代码。

（1）复制项目到本地。

在终端执行命令：

```
cd ~/desktop
git clone https://e.coding.net/ninghao/xb2/xb2_flutter.git xb2_flutter_done
```

上面两行命令可以把笔者做好的项目放到读者桌面的 xb2_flutter_done 目录下。

（2）查看提交。

查看项目的提交历史，可以使用带图形界面的 Git 工具。例如，用 Sourcetree 软件打开

笔者已完成项目的效果，如图 2.14 所示。

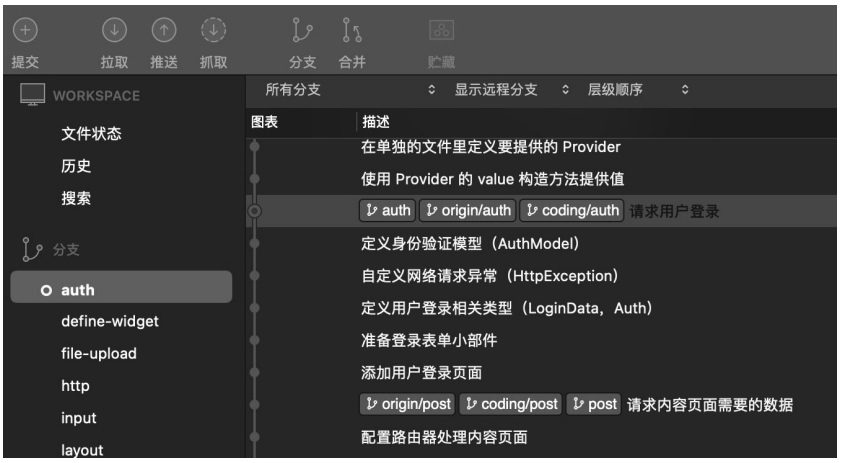

图 2.14　Sourcetree 软件截图

最初，左侧“分支”列表中只有一个 master 本地分支。选中 master 分支，观察右侧的提交历史，会发现有一些 origin 开头的分支，如 origin/auth，表示这是一个远程分支。双击远程分支，可以基于它创建一个本地分支，这样你在左侧“分支”列表里就可以看到 auth 这个分支了。

双击提交的描述，可以检出（checkout）这次提交，也就是把项目当前的状态暂时切换到这次提交保存的状态上。这时如果运行调试，用户看到的应用就是完成当前提交后的样子。

单击选中某次提交，页面下方会显示与该次提交相关的信息，如会显示这次提交修改了哪些文件，选中一个文件，右侧会显示在这次提交里该文件发生了怎样的变化（见图 2.15）。

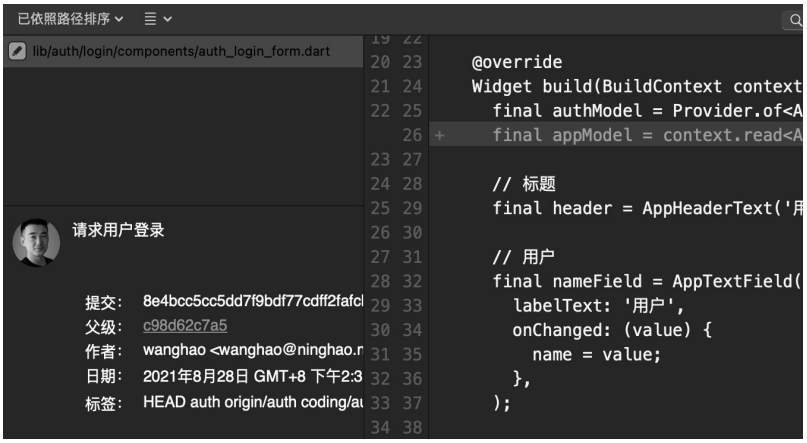

图 2.15　在 Sourcetree 软件里查看项目的提交历史记录

除了 Sourcetree 这种图形界面的 git 工具外，读者也可以直接通过 github 或 coding 网站浏览本书配套项目的提交历史。