第 3 章

Flutter 运行环境介绍

本章主要介绍 VS Code 开发界面、第 1 个程序的含义、开发过程中常用快捷键的使用、热加载、Flutter 程序相关的命令、程序的调试等。

3.1 界面的介绍

当单击桌面图标时，即可开始运行 VS Code 软件，如图 3-1 所示。

图 3-1 VS Code 桌面图标

启动后的界面如图 3-2 所示，主要包括最上面的菜单栏和最下面的状态栏及图中间的四个区域。

最左边标记为 A 的区域为侧边区，包含不同的视图，以便在处理项目时提供帮助。有资源管理器、搜索、源代码管理、运行和调试、插件的安装、测试和 Flutter 框架程序结构图。侧边区左下角为账户和管理功能。当鼠标悬浮到对应的图标时有相应的提示。

单击侧边区的资源管理器，可以展开或收缩右边的工程结构区域，从中可以看到项目工程结构(标记为 B 的区域)。

如图 3-3 所示，其中 lib 为 Flutter 框架 Dart 语言源代码所在区域，android 为生成安卓源代码的位置，ios 里面存放 ios 工程的所有内容，web 为和网页有关的源代码。pubspec. yaml 文件为工程配置文件，包含第三方插件的引入和标记本地资源(如音频、视频、图片、字体、JSON 文件等)的位置等。

中间最大的区域(标记为 C 的区域)为编辑区域，编辑主要的功能为编写 Dart 源文件。可以打开任意数量的编辑器，也可以并排垂直或水平排列多个窗口。

在编辑器区域下方显示了不同的选项卡面板(标记为 D 的区域)，可以显示输出、调试、错误或警告信息。在终端中，可以执行 DOS 命令，如查看 Flutter 运行环境、创建工程等。调试控制台用于显示程序运行时的各种输出信息，帮助我们观察程序运行时的状态。面板也可以左右移动，以获得更多的空间。

状态栏为项目和文件的提示信息，如鼠标的位置、是否有错误或警告信息等。在状态栏右边位置，包含 Dart DevTools(Dart 开发工具)、Flutter 版本信息，单击如箭头所示的位置可以启动安卓虚拟机，如图 3-4 所示。

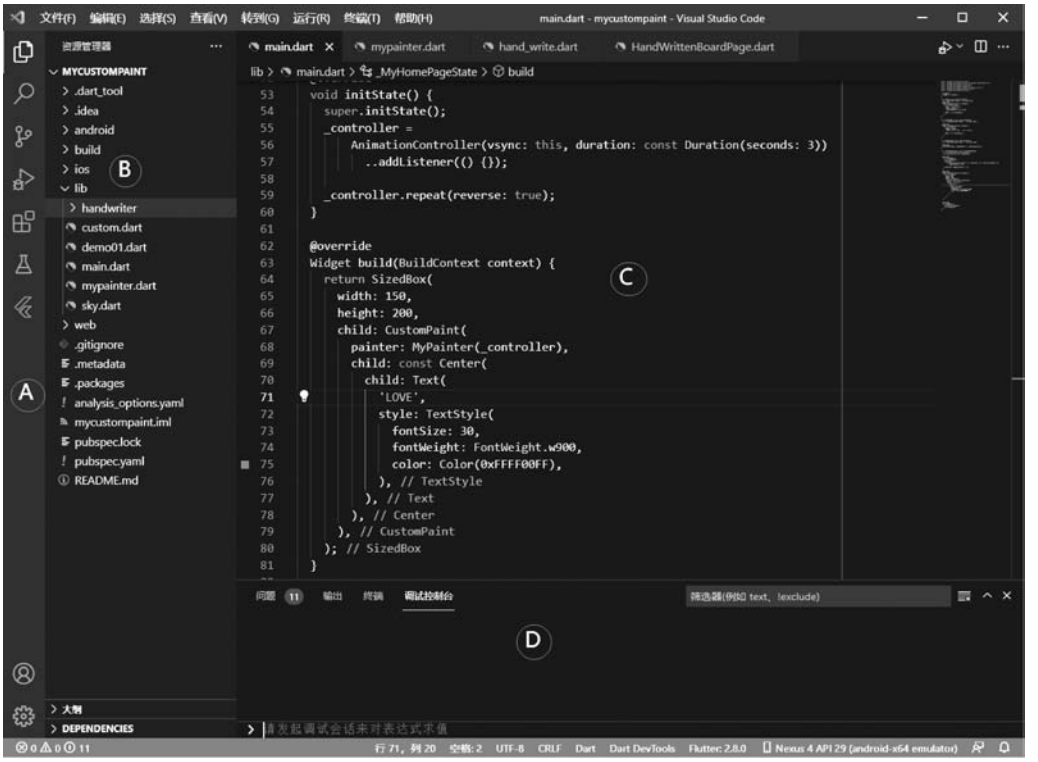

图 3-2　VS Code 启动界面

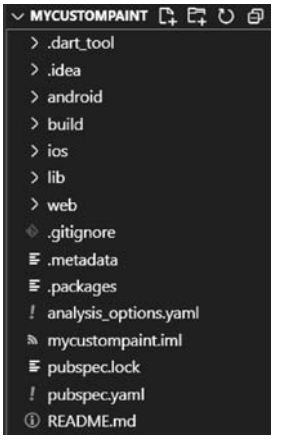

图 3-3　Flutter 工程结构图

图 3-4　状态栏信息

3.2 创建 Flutter 工程

在 DOS 提示符下，或单击 VS Code 菜单“终端”→“新终端”，可以在 G 盘下新创建一个文件夹，如 flutterdemo，并在文件夹中通过命令创建 Flutter 工程 myflutter，如图 3-5 所示。图中表示的含义依次如下。

（1）md flutter：通过 md 命令创建了一个文件夹 flutterdemo。

（2）cd flutter：通过 cd 命令进入了 flutterdemo 文件夹。

（3）flutter create myflutter：通过 flutter create myflutter 命令创建了一个名为 myflutter 的 Flutter 工程。

（4）cd myflutter：通过命令进入 myflutter 工程中。

（5）flutter run：运行 flutter 程序。

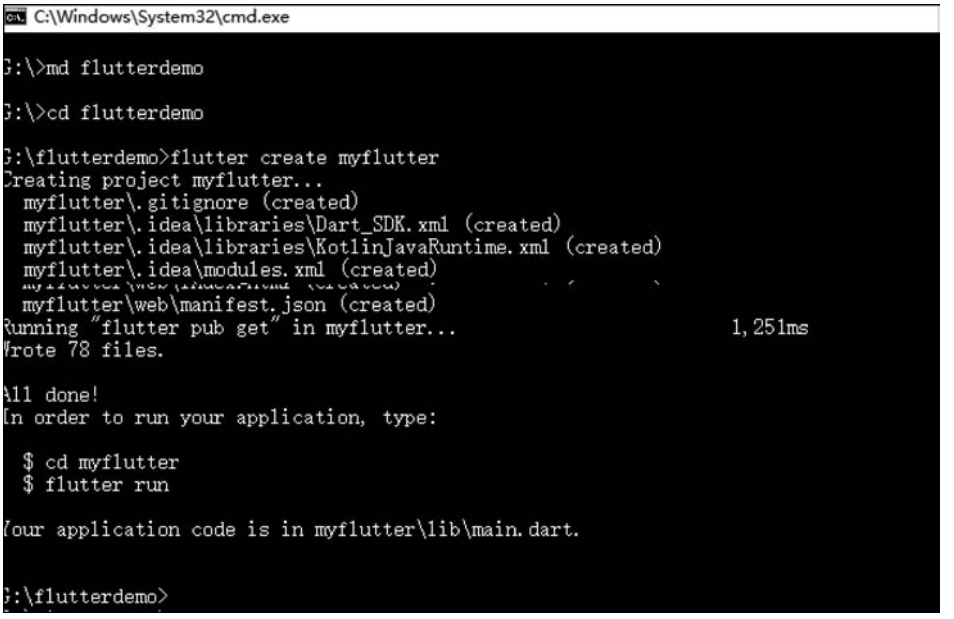

图 3-5 创建 Flutter 工程

在 VS Code 中，通过“文件”菜单中的“文件夹”，找到工程所在的位置（书中的位置为 G:\flutterdemo\myflutter），单击“选择文件夹”，即可在 VS Code 中打开 Flutter 工程。其中文件夹的名称为工程的名称，如图 3-6 所示。

打开的窗口界面如图 3-7 所示。在左边的侧边栏资源管理器中找到 lib 文件夹中的 main.dart 文件，双击此文件便可打开，在中间的编辑窗口会显示 main.dart 的源文件内容。

在中间的编辑窗口，删除所有的内容，输入的内容如下：

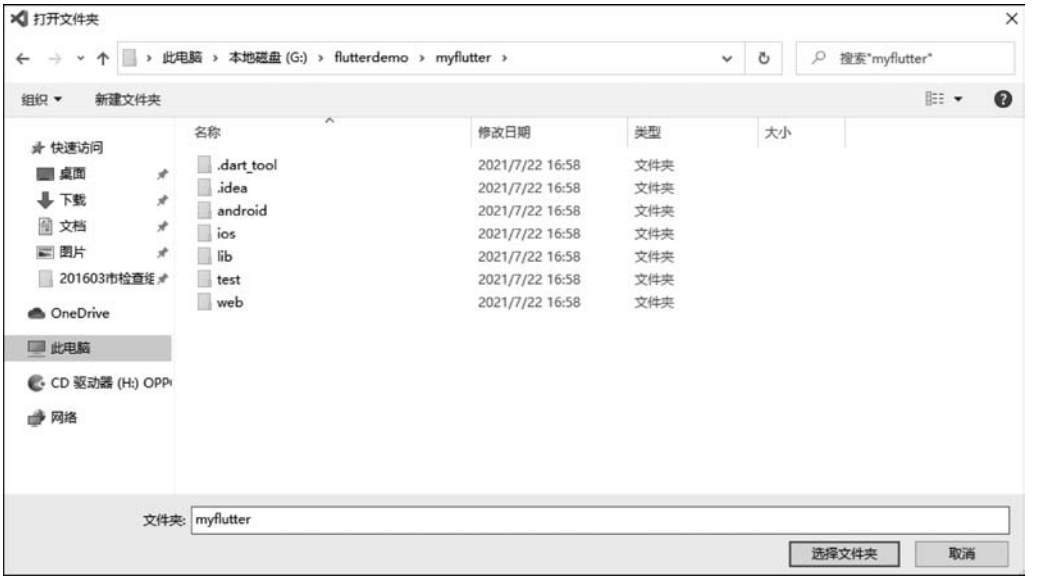

图 3-6　打开 Flutter 工程

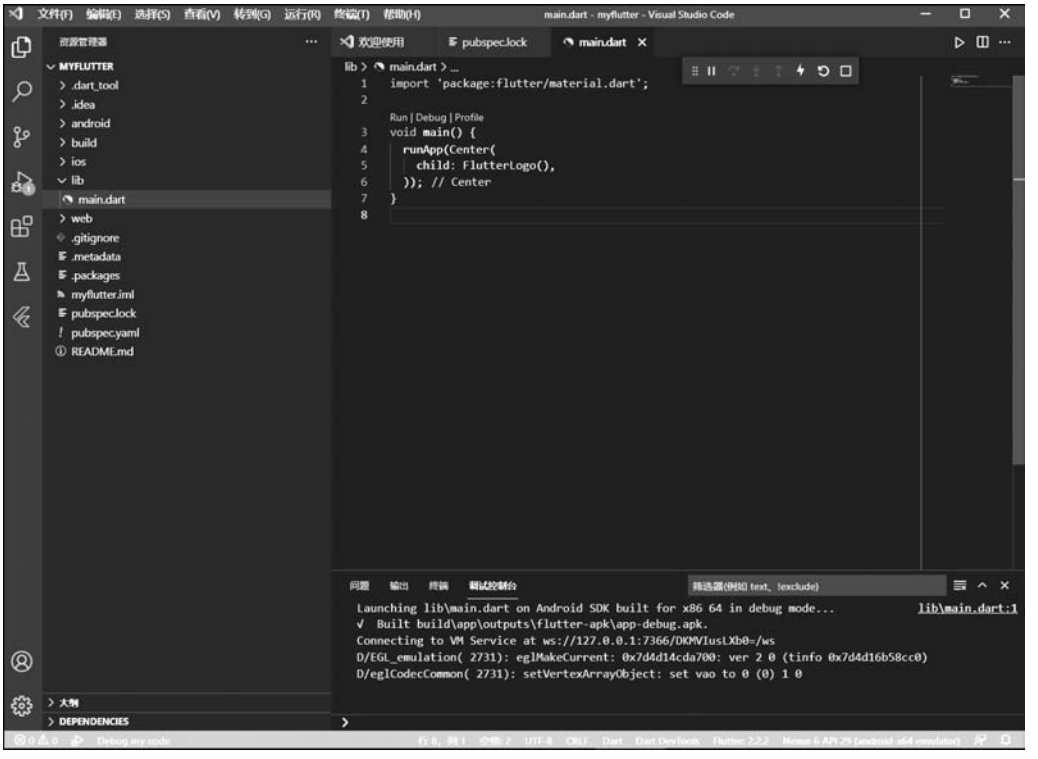

图 3-7　Flutter 工程结构

```
//第 3 章 3.1 FlutterLogo 图片的显示
import 'package:flutter/material.dart';          //-----------------第 1 行

void main() {                                    //-----------------第 2 行
  runApp(Center(                                 //-----------------第 3 行
    child: FlutterLogo(),                        //-----------------第 4 行
  ));
}
```

程序代码的含义如下：

第 1 行表示导入 flutter material. dart 包。material. dart 用于实现 Material Design 设计。

第 2 行表示程序的入口点 main，void 表示没有返回值。

第 3 行 runApp 表示将内容显示在屏幕上，Center 表示居中显示。

第 4 行表示 Center 的子属性 child 中包含 FlutterLogo()，FlutterLogo 是系统自带的图形。

最后括号也要成对地匹配。整个程序的含义表示在屏幕的中央显示一个 FlutterLogo 图形。如果侧边栏目中 test 文件夹有错误，则实际上是测试原来的内容，所以这里删除 test 文件夹。在最下边状态栏中，单击 no device 项，窗口上方会出现如图 3-8 所示可以运行 Flutter 的环境，选择已安装的虚拟机(这里是 Nexus 6 API 29 mobile)，就可以启动虚拟机了。

虚拟机启动后，在 VS Code 菜单中，单击“运行”→“启动调试”，就可以在虚拟机中安装 Flutter 程序了，最后显示的内容如图 3-9 所示，在屏幕的中央显示一幅图形。

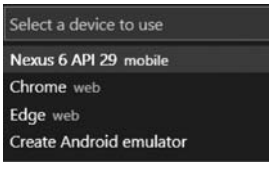

图 3-8　启动 Android 虚拟机

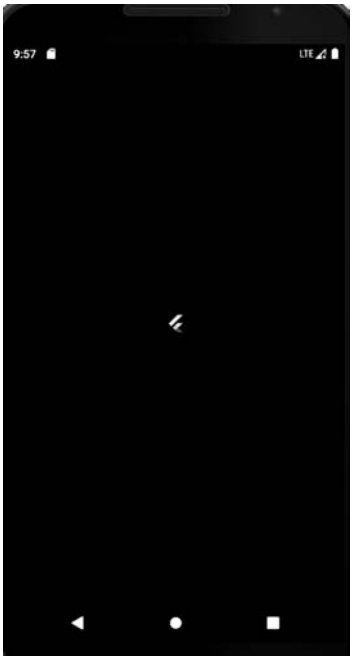

图 3-9　在屏幕的中央显示 FlutterLogo 图形

在 VS Code 编辑窗口右上角有浮动工具栏，如图 3-10 所示，图中闪电图标为热加载(Hot Reload)图标，Flutter 的热重载功能可以帮助我们在无须重新启动应用的情况下快速、轻松地进行测试、构建用户界面、添加功能及修复错误。当我们修改界面的内容时，可以在虚拟机上同步看到效果。当把鼠标放到图中的图标上时，会有相应的功能提示。

图 3-10　VS Code 中的浮动工具栏

浮动工具栏上从右到左依次为冷重启(方框图标)、热重启(刷新图标)和热加载(闪电图标)，如图 3-10 所示。

热加载主要用于虚拟机和组件树中组件的改变，以及保存应用的状态，而全局变量的初始化、静态变量的初始化、main 方法中的代码变化等则需要用热重启技术。冷重启费时更长，因为它要将代码重新编译成安卓、iOS 代码，在页面应用上，也需要重新开启 Dart 开发编译器。

恭喜你，现在已开启 Flutter 程序之旅了。

3.3　VS Code 中 Flutter 编辑技巧

3.3.1　组件提示功能

在 VS Code 主窗口中，当鼠标浮动在组件名称上方时，会有相应的提示，即提示这个组件有哪些属性及用法说明，如图 3-11 所示。可知 FlutterLogo 有 size 属性(也可以通过 Flutter api 文档查看 FlutterLogo 属性、方法和如何使用等信息)，这里 size 表示可以设置图形的大小。另外也可以通过 textColor 设置文本颜色，通过 style 设置风格，通过 curve 设置显示的动画风格，如提示中的 fastOutSlowIn 表示快出慢进。

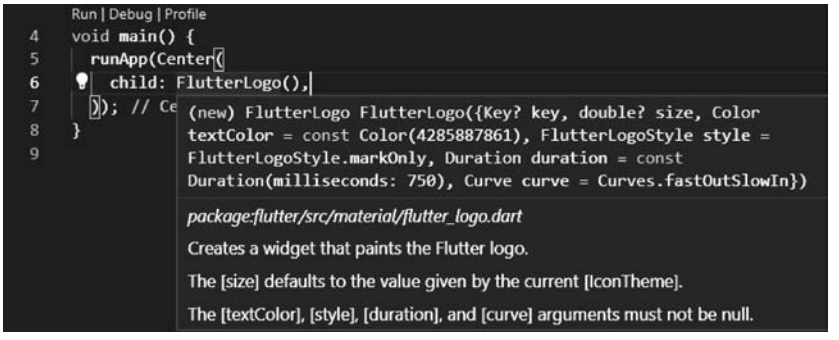

图 3-11　Flutter 组件说明

根据上面的提示，我们可以将图 3-9 显示的图形加上 size 属性，并将大小赋值为 96，如下面代码的第 4 行所示。

```
//第 3 章 3.2 FlutterLogo 更大尺寸图片的显示
import 'package:flutter/material.dart';          //-----------------第 1 行

void main() {                                    //-----------------第 2 行
  runApp(Center(                                 //-----------------第 3 行
    child: FlutterLogo(size:96),                 //-----------------第 4 行
  ));
}
```

保存后，由于自动使用了热加载技术，所以图 3-9 虚拟机屏幕中的图形将会自动变大。

3.3.2 在 Flutter 工程中插入和提取组件

当在 VS Code 编辑窗口中单击组件名称时，在左边会出现一个黄色的灯泡图标，如图 3-12 所示。单击灯泡图标，弹出的菜单分别表示的含义如下。

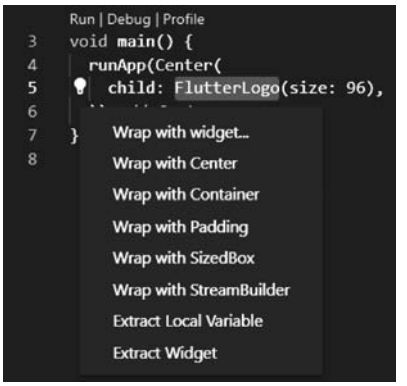

图 3-12 对组件的封装和提取

(1) Wrap with widget：在外边再包裹一个组件，单击后，可以更改为自己想要的名称。

(2) Wrap with Center：在外边包裹一个居中组件，使里面的组件居中显示。

(3) extract Local Variable：扩展成一个本地变量。

(4) extract Widget：扩展成一个独立组件，便于以后可以重复使用这段代码。

3.3.3 自动导入包

当添加新的组件时，如果没有导入相应的包，则会有错误提示。这时，可将鼠标浮于组件上方，单击快速修复，选择 import 'package:flutter/material.dart';，则可以自动导入相应的包，如图 3-13 所示。

3.3.4 快捷键的使用

在 VS Code 编辑器中，如输入 stl，则表示可以自动生成无状态组件，st 表示自动生成有

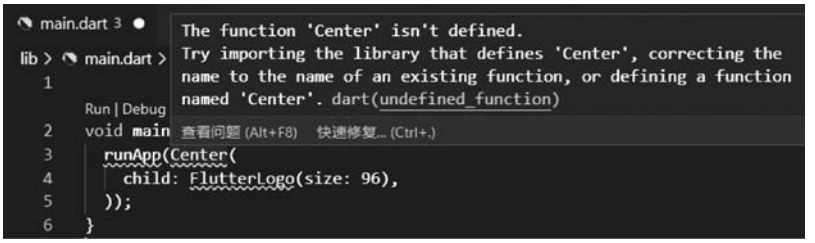

图 3-13　自动导入 Flutter 库文件

状态组件。输入 staim 则可以自动生成一个带有动画的有状态组件，如图 3-14 所示。这样可以省略敲很多代码，自动生成的代码格式也比较工整。

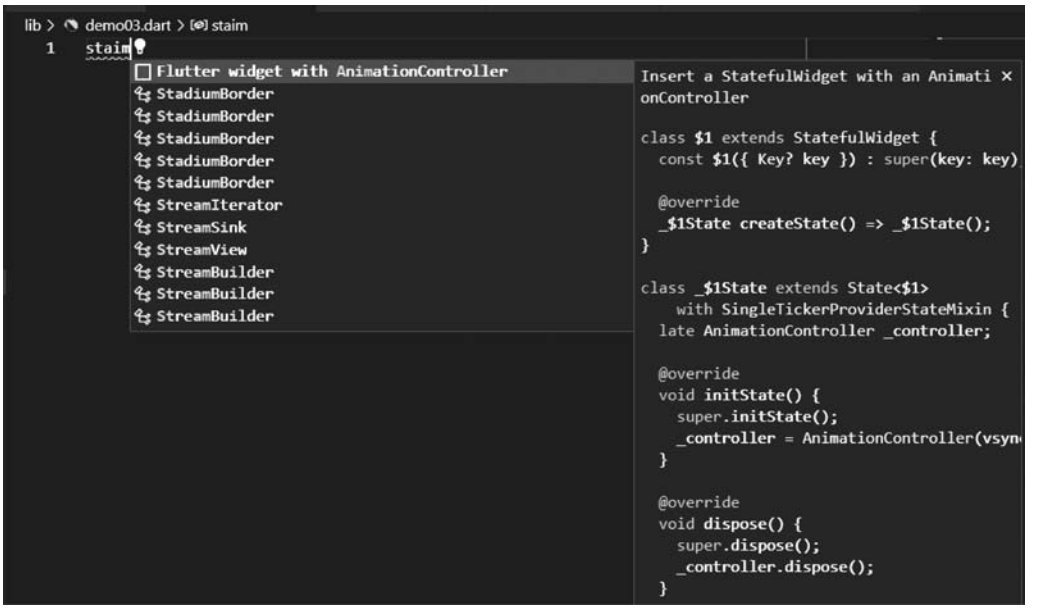

图 3-14　使用快捷方式生成带动画的有状态组件

常用的快捷键有很多，如 Ctrl＋/表示注释这一行，Ctrl＋F5 表示热重启。用户可以根据自己的喜好进行选择。

3.4　Flutter 项目的分析与调试

3.4.1　Flutter 项目分析

可以使用 flutter analyze 命令对整个 Flutter 工程项目的 Dart 代码进行分析，例如可以发现如下问题，如多余的导入类、变量使用的不规范、过期属性的使用、调用类中出现了问题等。使用方式为在工程的命令行中输入命令 flutter analyze，结果如下：

```
PS C:\flutterproject\moodinn > flutter analyze
Analyzing moodinn...
info - The value of the field '_favoritesDB' isn't used -
         lib\screens\me\favorites.dart:17:15 - unused_field
info - 'accentColor' is deprecated and shouldn't be used. Use
          colorScheme.secondary instead. For more information, consult the migrationguide at
https://flutter.dev/docs/release/breaking - changes/theme - data - accent - properties #
migration - guide. This feature was deprecated after v2.3.0 - 0.1.pre. -
         lib\screens\home.dart:81:40 - deprecated_member_use
info - Unused import: 'package:moodinn/db/db_helper.dart' -
         lib\screens\me\favorites.dart:2:8 - unused_import
```

在上面的结果中，可以看出有以下的提示信息，'_favoritesDB'变量定义后，没有被使用，使用了过期的'accentColor'属性，db_helper.dart 导入了，但是没有被使用。我们可以根据提示信息进行相应修改。

3.4.2 程序的调试

最常用的调试方式为在开发过程中加入 print()语句，对变量的值进行输出，通过对输出结果的判断来分析程序中的问题，但是使用后，一定要注释或删除这条语句，以免影响程序的性能。也可以使用 deBugPrint 语句将消息打印到控制台，使用 Flutter 工具的“日志”命令(flutter logs)访问该消息，以避免在 Android 上的调试数据丢失。

另外的方式为在程序中加入断点，通过对程序的调试来发现问题，如图 3-15 所示。

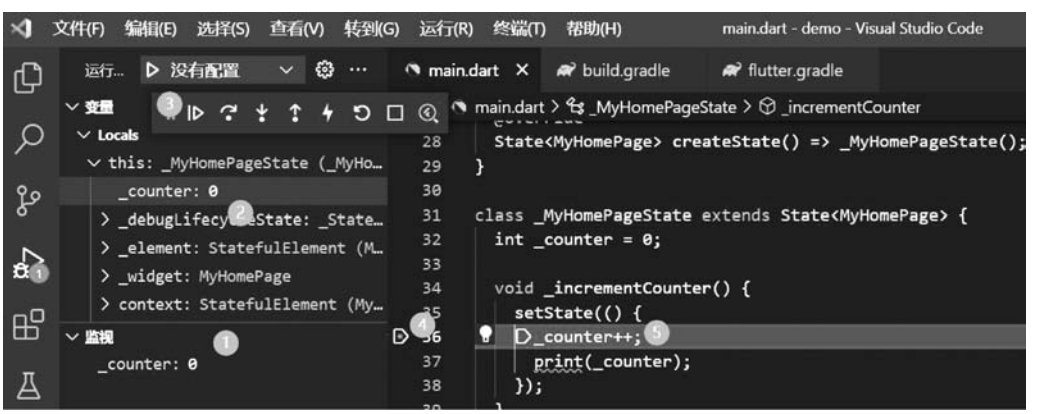

图 3-15 程序的调试

图 3-15 中，序号分别表示的含义如下：

(1) 表示在程序的调试过程中，观察指定变量的值的变化情况。

(2) 表示在程序的调试过程中，观察本地变量的值的变化情况。

(3) 表示可以使用单步调试的方式观察程序的执行情况，工具栏中的 分别表

示为程序的暂停(继续)、单步跳过、单步跳入和单步跳出。

(4) 表示在程序中加入的断点。

(5) 表示程序断点处的程序片断。

3.4.3　断言表达式

可以在开发调试环境(Debug)下加入 assert 断言语句,它使用布尔条件进行测试。如果条件不满足,则将显示断言错误,当条件满足时继续执行。assert 断言语句只能在开发模式下使用,而在生产发布环境(release)下将被忽略。

在以下的程序中,由于字符串不相等为假,所以程序执行时将会出现断言错误,如图 3-16 所示。

```
void main() {
  String name = "Jack";
  assert(name != "Jack", "如果字符串不相等,则显示此消息!");
}
```

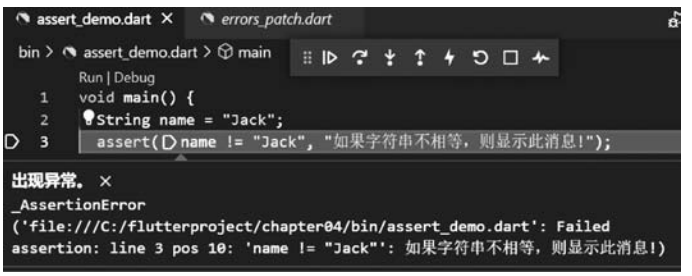

图 3-16　程序运行中的断言错误