序

Flutter/Dart 是跨平台 App 開發的後起之秀，但是短短數年，就成為使用率最高的解決方案。原因很簡單，第一是它能夠完整橫跨 Android、iOS、Windows、Mac、Linux 和 Web，放眼天下，無人能出其右。第二是它的執行速度和各平台的原生 App 一樣快，你無須擔心使用者體驗不佳的風險。第三是它有一個強大的靠山-Google，因此保證你不會成為技術孤兒！

這次改版有三個重點：

1. 加入 Dart 語言最新的 Null Safety 語法。
2. 用 ValueNotifier 和 ValueListenableBuilder 取代 StatefulWidget 搭配 GlobalKey 的作法，程式碼更簡潔。
3. 加入資料庫、Google 地圖和定位等技術主題。

除了上述三項，還有許多內文解說和圖檔的更新，目的就是要提高學習的流暢度和完整性。

Flutter/Dart 是比較新的技術。對於初學者來說，靠著網路上的片段資訊自行摸索，不僅學習效果不彰，而且很容易踩雷。再者，Flutter App 程式架構和各平台的原生程式有很大的差異，一開始就會用到物件導向技術和語法，因此需要先建立相關基礎，才能夠了解程式的架構。

基於對初學者需求的考量，本書從 Flutter App 開發的實務面著手。先用最簡單的範例帶入基本觀念和 Dart 語言基礎，並藉由操作步驟講解，幫助讀者熟悉 Android Studio 的使用技巧。接著由淺入深，依序學習各項主題。每一個章節的內容都能夠承先啟後，讓學習可以順利地延續。另外在講解的過程中，會適時搭配 Dart 語法介紹和範例，讓讀者同時兼顧 Flutter 和 Dart 的學習，無須因為不熟悉 Dart 語言而放棄學習 Flutter。

學習程式設計最好的方式就是親自動手，從無到有把專案完成。實作的過程一定會出現疑問和錯誤，這也是學習的機會。如果可以自己找到答案和排除錯誤，就表示你的功力和經驗又向前邁進了一步。如此經年累月持續下去，自然就能夠累積實力。坐而言不如起而行，現在就讓我們開始動手吧！

孫宏明

ElevatedButton、Toast 和 SnackBar

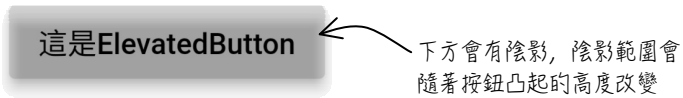
09

學習重點

1. 使用按鈕。
2. 學習 Lambda 函式。
3. 顯示 Toast 和 SnackBar 訊息。

前面完成的 App 只能夠顯示文字和影像，無法讓使用者操作。雖然它的功能很簡單，但是我們因此學會了 Flutter App 的架構和一些基礎物件，以及 Dart 的基本語法。接下來要進入一個新的階段，我們要完成一個可以讓使用者操作的 App。

9-1 ElevatedButton



▲ 圖 9-1 ElevatedButton 範例

ElevatedButton 字面上的意思是「凸起的按鈕」（參考圖 9-1），它的邊緣會有陰影效果。我們可以利用參數控制它凸起的高度。凸起的高度愈高，陰影範圍愈大。按鈕和我們之前學過的物件的最大差異是它可以讓使用者操作。當使用者按下按鈕時，App 會執行對應的功能。

要讓按鈕執行某一項功能需要用到單元 3 學過的函式。只要把函式設定給按鈕，當該按鈕被按下時，就會執行該函式。只不過設定給按鈕的函式可以做一些簡化。我們在單元 3 介紹的函式有函式名稱，這個函式名稱是用來呼叫該函式（也就是執行該函式）。但是設定給按鈕的函式是該按鈕專用，我們不會在其他

地方呼叫這個函式，因此可以把函式名稱省略。這種沒有函式名稱的函式就叫做「Lambda 函式」，它的格式如下：

```
(函式的參數) {          ← Lambda 函式沒有函式名稱

    函式裡頭的程式碼

}
```

讀者可以和單元 3 的函式格式比對，Lambda 函式少了傳回值型態和函式名稱這二個部分。Lambda 函式的使用時機是，如果該函式只會在程式碼中出現一次，就可以使用 Lambda 函式。

我們來看一個 ElevatedButton 搭配 Lambda 函式的範例：

```
var appBody = Container(
  child: ElevatedButton(
    child: Text('我是按鈕',),
    onPressed: () {
      // 按下按鈕後要執行的程式碼
   },
  ),
  alignment: Alignment.topCenter,
);
```

建立一個 ElevatedButton 物件，直接設定給 Container 的 child 參數

這個範例用到 ElevatedButton 的 child 參數和 onPressed 參數。child 參數是設定按鈕上顯示的文字，onPressed 參數必須設定一個函式，這個函式是按下按鈕後要執行的程式碼。我們可以設定一個一般函式，這裡是用 Lambda 函式。

onPressed 參數接收的函式不可以有參數，所以這個 Lambda 函式的圓括弧裡頭是空的。ElevatedButton 可以使用 style 參數設定按鈕的外觀，我們可以利用 ElevatedButton 的 styleFrom()方法來設定，以下是 styleFrom()方法常用的參數：

1.primary

設定 ElevatedButton 的顏色，指定顏色的方式請參考單元 6 的說明。

2. elevation

設定按鈕凸起的高度。

3. padding

   設定按鈕文字與按鈕邊緣的距離，總共有上下左右四個方向，必須用 EdgeInsets 類別來設定。

稍後會有實際的使用範例。接下來我們要讓按鈕按下後，在 App 畫面上顯示一段 Toast 訊息，因此要先學會如何使用 Toast。

9-2 使用 Toast 套件

Toast 的功能是在 App 畫面顯示一段文字，這段文字只會持續幾秒鐘，之後就會消失。要使用 Toast 需要載入額外的套件。我們以上一個小節的 ElevatedButton 為例，幫它加入顯示 Toast 訊息的功能。

step1 在 Android Studio 左邊的專案檢視視窗找到專案設定檔 pubspec.yaml。將它開啟，找到其中的「dependencies:」區塊，在該區塊最後加入以下粗體字的程式碼：

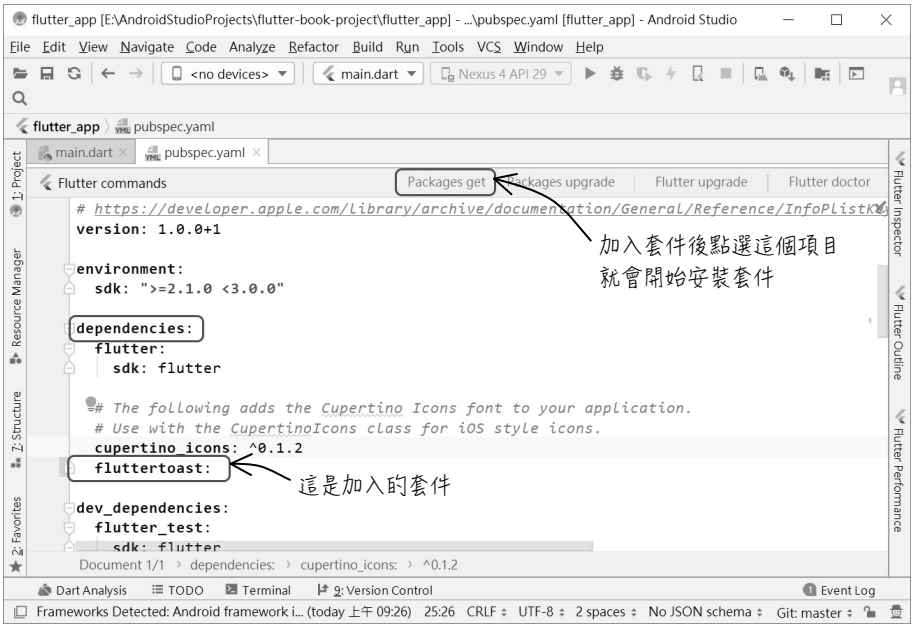
```
...（其他程式碼）

dependencies:
  ...（原來的程式碼）
  fluttertoast:          注意最後有一個分號「:」

...（其他程式碼）
```

我們加入的是 Toast 套件的名稱。套件名稱後面是一個分號，分號後面可以指定套件的版本號碼，例如「fluttertoast: ^3.1.3」。版本號碼前面的「^」符號表示比這個版本新的都可以使用。如果不需要指定版本，就把分號之後留白。

step2 加入套件之後編輯視窗上方會出現一行指令，如圖 9-2。點選 Packages get 就會開始載入套件，Android Studio 下方會出現一個視窗，顯示執行過程和結果。



▲ 圖 9-2　加入套件的操作畫面

step 3 開啟程式檔，在第二行加入下列粗體字的程式碼，它用 import 指令載入套件中的程式檔。另外我們用前一小節介紹的參數設定 ElevatedButton 的顏色、凸起的高度和文字到邊緣的距離。最後在設定給 ElevatedButton 的 Lambda 函式中，呼叫 Fluttertoast 類別的 showToast()方法顯示訊息。msg 參數是設定要顯示的訊息，toastLength 參數是設定訊息停留的時間長度，它有長和短二種選項。gravity 參數是控制訊息顯示的位置，它有螢幕上方、中央和下方三種選項。backgroundColor 是設定訊息的背景顏色，textColor 是設定文字顏色，fontSize 是設定文字大小。

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

void main() => runApp(const App());

class App extends StatelessWidget {
  const App({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // 建立 appTitle 物件
    var appTitle = const Text('Flutter App');
```

用 import 指令載入套件

```
    // 建立 appBody 物件
    var appBody = Container(
      child: ElevatedButton(
        child: const Text(
          '顯示 Toast 訊息',
          style: TextStyle(fontSize: 20, color: Colors.redAccent,),
        ),
        style: ElevatedButton.styleFrom(
          primary: Colors.yellow,
          padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 20),
          elevation: 8,
        ),
        onPressed: () {
          // 顯示 Toast 訊息
          Fluttertoast.showToast(msg: '你按下按鈕',
              toastLength: Toast.LENGTH_LONG,
              gravity: ToastGravity.CENTER,
              backgroundColor: Colors.blue,
              textColor: Colors.white,
              fontSize: 20.0);
        },
      ),
      alignment: Alignment.topCenter,
      padding: const EdgeInsets.all(30),
    );

    // 建立 appBar 物件
    var appBar = AppBar(
      title: appTitle,
    );

    // 建立 app 物件
    var app = MaterialApp(
        home: Scaffold(
      appBar: appBar,
      body: appBody,
    ));

    return app;
  }
}
```

設定 ElevatedButton 的顏色、凸起的高度和文字到邊緣的距離

顯示 Toast 訊息，我們可以用參數控制訊息要如何顯示

編輯好程式檔之後啟動執行，然後按下畫面上的按鈕，就會看到圖 9-3 的結果。

何謂套件

套件又叫做程式庫，它是已經寫好的函式或類別，我們可以利用它們來執行某些功能，像是 ElevatedButton、Text、Center、Container...都是套件裡頭的類別。有些套件是系統內建的，我們可以直接使用。有些套件必須另外安裝，才能夠載入專案中使用。

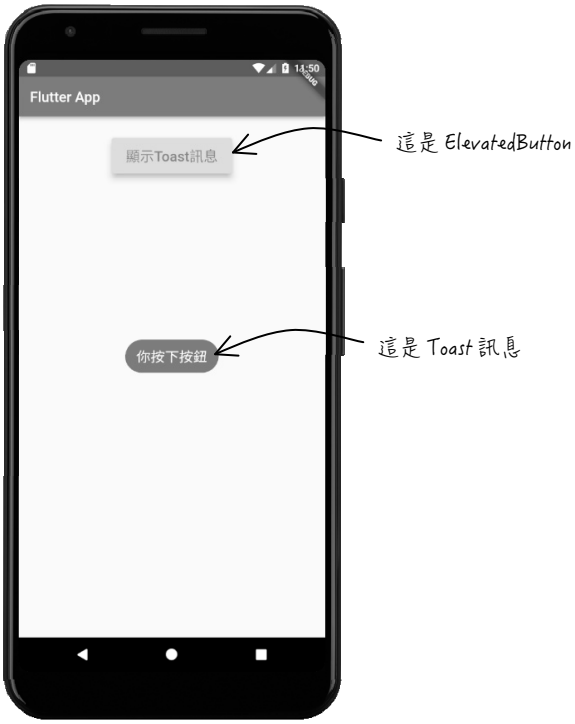


圖 9-3　按下按鈕顯示 Toast 訊息的畫面

9-3 顯示 SnackBar 訊息

SnackBar 的功能和 Toast 類似，也是用來顯示訊息，只不過訊息出現的位置不一樣，型態也不一樣。SnackBar 訊息還可以加入點選的功能，不過 SnackBar 必須搭配 Scaffold 物件才能運作。

我們用圖 9-4 解釋 SnackBar 和 Scaffold 物件的關係。圖 9-4 是以前一小節的程式範例為基礎，然後新增一個 AppBody 類別，我們要用它來取得 Scaffold 物件。圖 9-4 展現了物件之間的關係，括號裡頭是它的類別。最上層是一個 App 類別的物件，它繼承 StatelessWidget 類別。

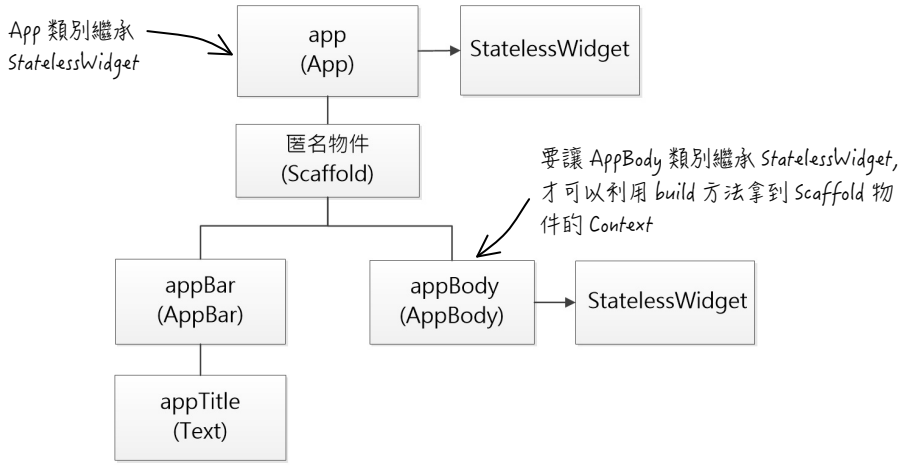


圖 9-4　SnackBar 範例 App 架構圖

圖 9-4 右下方是 appBody 物件，它是程式畫面。如果要在 appBody 中使用 SnackBar，必須拿到上層 Scaffold 物件的 Context。Context 是系統傳給程式的資源，我們可以利用 build()方法的參數取得 Context，就像 App 類別的 build()一樣，所以只要讓 appBody 繼承 StatelessWidget，就可以利用 build()方法的參數得到 Scaffold 物件的 Context。根據以上討論，我們將程式檔修改如下：

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

void main() => runApp(const App());

class App extends StatelessWidget {
  const App({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // 建立 appTitle 物件
    var appTitle = const Text('Flutter App');
```

```
            textColor: Colors.white,
            onPressed: () =>
                Fluttertoast.showToast(msg: '你按下 SnackBar',
                    toastLength: Toast.LENGTH_LONG,
                    gravity: ToastGravity.CENTER,
                    backgroundColor: Colors.blue,
                    textColor: Colors.white,
                    fontSize: 20.0),
          ),
        );

        // 顯示 SnackBar
        ScaffoldMessenger.of(context).showSnackBar(snackBar);
      },
    ),
    alignment: Alignment.topCenter,
    padding: const EdgeInsets.all(30),
  );

  return widget;
 }
}
```

這裡要用 context 來顯示 SnackBar 訊息

這段程式碼最大的改變是把 appBody 物件換成用新的 AppBody 類別產生，而且 AppBody 類別是繼承 StatelessWidget。我們從 build() 方法的 BuildContext 參數取得 Context。build()方法中的程式碼就是原來產生 appBody 物件的程式碼，當使用者按下 ElevatedButton 時，我們就建立一個 SnackBar 物件，然後將它顯示出來。SnackBar 有以下參數可以使用：

1. content

   要顯示的文字。

2. duration

   訊息停留的時間長度，必須利用 Duration 類別設定。

3. backgroundColor

   設定訊息的背景顏色。

4. shape

   設定訊息背景的形狀，可以用 CircleBorder、RoundedRectangleBorder... 來設定。

5. action

在 SnackBar 中加入點選的功能，它的運作方式類似按鈕，只不過名稱叫做 SnackBarAction。它的 label 參數是設定點選的文字，textColor 是設定文字顏色，onPressed 參數就如同 ElevatedButton 的 onPressed 參數一樣，必須設定一個 Lambda 函式。上面的範例是設定點選之後會顯示一個 Toast 訊息。

建立 SnackBar 物件之後，只要利用 ScaffoldMessenger 的 showSnackBar() 方法就可以顯示 SnackBar。圖 9-5 是程式的執行畫面。

圖 9-5　SnackBar 範例

10 其他型態的按鈕

學習重點

1. 改善程式碼的架構。
2. 類別的公開和私有方法。
3. 更多型態的按鈕。

除了像 ElevatedButton 這種凸起式按鈕之外，還有其他不同類型的按鈕可以使用。本單元要介紹其他型式的按鈕。不過在開始之前，我們要改良一下程式的架構，讓它更容易理解和維護。

10-1 改良程式架構

程式的功能愈多，程式碼自然就會愈來愈長。Flutter App 的程式碼有一個很明顯的特點，就是它是由許多物件一層一層架構起來的。上一個單元圖 9-4 就呈現出這樣的特性。這樣的架構反應在程式碼，就變成物件的參數是另一個物件，所以括弧會有很多層，程式碼會呈現多層次的內縮，物件和它所屬的參數會變得很難對應。

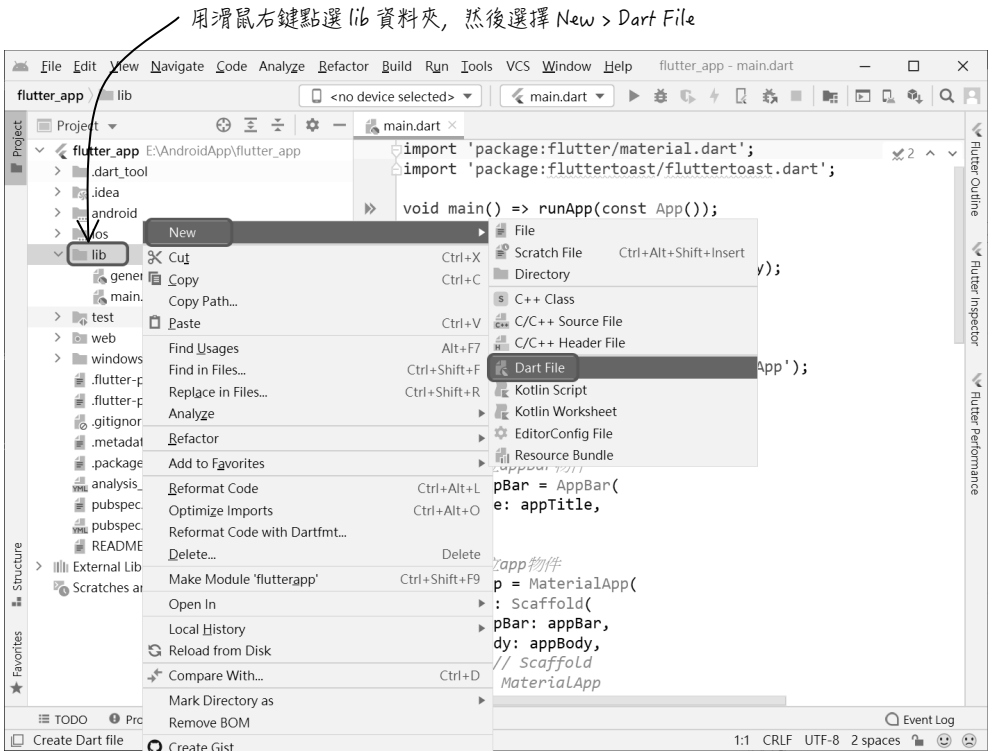
要改善這個問題有三種做法：

1. 先把物件建立好，再傳給另一個物件的參數。就像之前的程式範例，我們先建立 appBar 和 appBody 這二個物件，再將它們設定給 MaterialApp 的 appBar 參數和 body 參數。
2. 如果參數是接收一個 Lambda 函式，而且 Lambda 函式的程式碼不是只有短短一、二行。這時候可以把 Lambda 函式的程式碼寫成一個方法，然後在 Lambda 函式裡頭呼叫這個方法。

3. 當程式愈來愈長，我們可以將程式碼做適當的分割，把它們搬移到不同的程式檔。

現在我們就運用上述技巧來改良前一個單元的程式碼。

step1 在 Android Studio 左邊的專案檢視視窗中展開專案，用滑鼠右鍵點選 lib 資料夾，然後選擇 New > Dart File（參考圖 10-1）。



▲ 圖 10-1　新增 Dart 程式檔

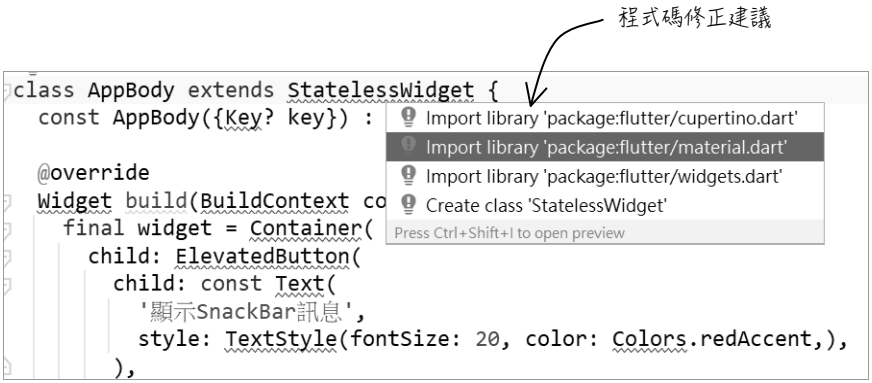
step2 在檔名對話盒輸入 app_body，按下 Enter 鍵。

命名 Dart 程式檔

Dart 程式檔應該用小寫英文字母和底線字元來命名，例如 app_data.dart。除了程式檔，Dart 的套件、程式庫和資料夾也應該用這種方式命名。

step3 新增的程式檔會顯示在編輯視窗。我們把主程式檔 main.dart 裡頭的 AppBody 類別的程式碼剪下來，貼到新程式檔 app_body.dart。

step4 程式碼會出現許多紅色波浪底線，表示有語法錯誤。這是因為我們還沒有載入需要的套件。我們可以利用程式碼輔助功能幫我們載入這些套件，先把編輯游標設定到第一個紅色波浪底線裡頭。然後先按住鍵盤的 Alt 鍵，再按下 Enter 鍵，就會顯示一個建議視窗（參考圖 10-2）。



▲ 圖 10-2　用 Alt 和 Enter 鍵啟動程式碼修正建議

step5 在程式碼修正建議中選擇載入 material.dart，也就是和原來的程式檔使用一樣的套件。程式碼大部分的紅色波浪底線都會消失，只剩下 Fluttertoast 的部分，因為它需要另一個套件。我們可以用同樣的操作技巧載入 fluttertoast.dart。

step6 接下來要修改程式碼，把按下按鈕要做的事寫成一個方法，然後在 onPressed 的 Lambda 函式中呼叫這個方法。以下是修改後的結果，粗體字是有更動的部分。

```
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';

class AppBody extends StatelessWidget {
  const AppBody({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final widget = Container(
      child: ElevatedButton(
        child: const Text(
```

類別的公開（public）和私有（private）方法

我們已經學過，類別的方法就是用來處理類別內部資料的函式。類別的方法有些可以讓外部程式使用，有些只可以在類別內部使用。前者稱為「公開方法」（Public Method），後者稱為「私有方法」（Private Method）。Dart 語言是用方法的名稱來區分公開和私有方法。如果方法名稱是用底線字元開頭，該方法就是私有方法，否則就是公開方法。不過 Dart 語言對於私有方法的限制比較不嚴格。只要在同一個程式檔裡頭，不同類別之間還是可以使用對方的私有方法。如果是不同的程式檔，就只能使用類別的公開方法。

step 7 切換到主程式檔 main.dart，裡頭會出現紅色波浪底線，因為 AppBody 類別已經搬到另一個程式檔。我們可以再次利用步驟 4 的操作技巧載入 app_body.dart。

現在可以啟動 App，測試功能是否正常。在改良程式架構的過程中，我們學會新的程式語法，以及如何使用程式碼修正建議。這些都是重要的基礎，後續會經常用到，請務必熟練。

10-2 各式各樣的按鈕

除了 ElevatedButton 之外，還有其他外觀不一樣的按鈕，請讀者參考圖 10-3。從按鈕名稱大概就可以知道每一種按鈕的特色：

1. ElevatedButton

   這是我們已經學過的凸起式按鈕。

2. TextButton

   以純文字為主的按鈕。

3. OutlinedButton

   它會顯示一個外框，底色是透明的，我們可以設定外框顏色和線條的樣式與粗細。

4. IconButton

   按鈕上會顯示一個圖示。

5. FloatingActionButton

   它的外觀和 ElevatedButton 類似，除了可以顯示文字之外，也可以顯示圖示。

6. ElevatedButton.icon

   這是 ElevatedButton 的一種變化，它可以在文字前面顯示一個圖示。

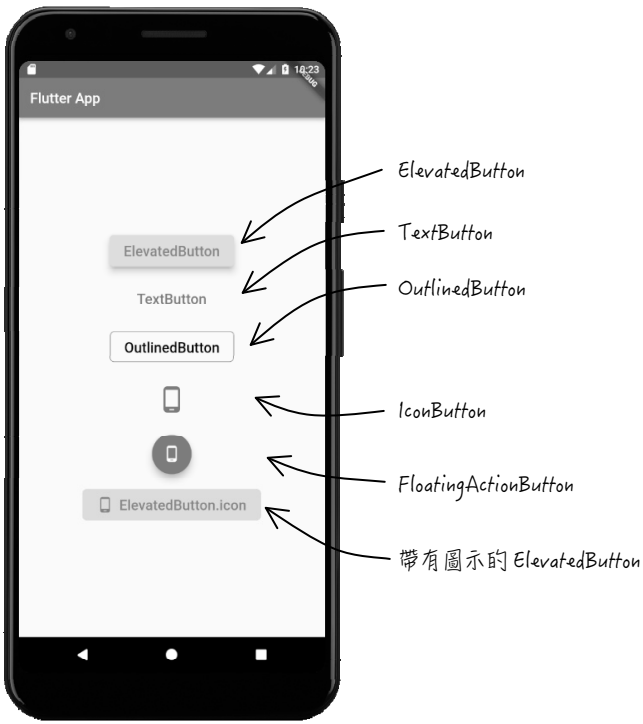


圖 10-3　不同類型的按鈕

以上按鈕有些部分的功能是類似的。也就是說，透過參數的設定，可以讓不同類型的按鈕呈現出類似的效果。例如 ElevatedButton、TextButton 和 OutlinedButton 都有一個 style 參數，這個參數是用來控制按鈕的外觀，像是顏色、文字和邊緣的距離、外型、高度...。這個參數的設定值可以用按鈕本身的 styleFrom()方法建立，就像上一個單元的 ElevatedButton 的做法一樣。