Chapter 10

程式偵錯與程式碼管理

大家應該都有過這樣的經驗，應用程式突然顯示執行異常訊息或沒回應，我們稱這種狀況爲「臭蟲（bug）」，而尋找臭蟲並修正錯誤的過程則稱爲「除錯（debug）」。

本單元說明如何應用Eclipse除錯功能來輔助我們分析程式錯誤原因並修正，內容包括：

- 了解語法錯誤、執行時期錯誤、邏輯錯誤的定義
- 解讀錯誤訊息，熟悉除錯工具
- 在程式中設定中斷點，並使用單步執行追蹤程式
- 監看程式執行時變數的內容
- 使用try…catch程式架構
- 使用Snippets管理程式碼

10.1 程式錯誤種類及原因

大多數的 bug 是因為程式設計考慮不夠周詳所致，當然功能愈複雜的程式，程式中潛藏錯誤的機率也相對較高。所以發佈應用程式前一定要經過一連串測試，而除錯也就成為開發應用程式必經之路，也因此有人說程式設計能力是從除錯中培養出來的，而如何善用除錯工具並熟練除錯技巧，自然成為程式設計訓練課程中相當重要的課題。

10.1.1 語法錯誤（Syntax Errors）

每種程式語言都有其語法規範，當程式敘述不符合語法規範時，該行程式敘述就存在語法錯誤。例如：遺漏標點符號、括弧使用不當、變數命名不正確（與保留字相衝突）、找不到元件、未 import 所需的套件等。由於初學者對程式架構、語法規則、指令用法...等較不熟悉，因此，在撰寫程式時也較容易疏忽而造成語法錯誤。由於語法錯誤都是發生在撰寫程式時，也稱為設計階段錯誤（Design-time Errors）。

專業的應用程式開發工具都具備即時的語法檢查功能，在程式編輯過程中，一旦發現程式敘述有語法錯誤，就會立即標示錯誤處，只要將滑鼠指到錯誤的位置，再依提示的錯誤訊息進行修正，一般而言，語法錯誤並不難處理。

Eclipse 會即時進行程式語法檢查，一旦程式碼語法有疑義，就會在語法疑義處加上紅色波浪底線，並在該行程式碼前標示紅色打叉記號。例如圖 10-1 程式碼中，findViewById()方法傳回型態為 View，而 txtmsg 為 TextView 型別，因此會標示錯誤。將滑鼠移到語法錯誤處，Eclipse 就會提示「Type mismatch: cannot convert from View to TextView」錯誤，並列出二種建議修正方式，我們只要使用滑鼠點選第一項，就可自動修正錯誤。

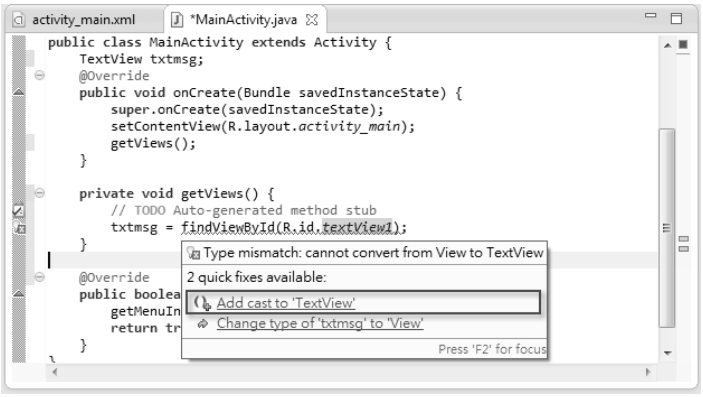

圖 10-1 Eclipse 自動檢查語法錯誤

10.1.2 執行時期錯誤（Run-time Errors）

即使每一行程式敘述語法都正確，並順利編譯建立執行檔，並不代表程式就沒問題。當程式執行時發生了系統能辨識的「Trappable Error」，我們說這個程式存在執行時期錯誤。例如：還未取得介面元件參照，即調用元件之方法、要儲存資料時磁碟空間不足、所要配置的記憶體空間不夠、或作業權限不符...等都會產生執行時期錯誤。

經驗豐富的程式設計師會考量程式執行時可能遇到的各種狀況，而在程式中加上一些特定狀態查核措施，以避開可能造成執行錯誤之狀況；或在程式中加上異常處理程序，以便在發生執行時期錯誤時，能檢知錯誤訊息，並指示使用者進一步處理。

try...catch 架構是當前程式設計經常用來防範執行時期錯誤的一種機制，詳細用法請參閱 10.3 節。

10.1.3 邏輯錯誤（程式產生非預期結果的錯誤）

邏輯錯誤是指程式的作業邏輯不正確而造成的錯誤，例如除法運算時，忽略了餘數的處理，雖然程式碼可能在毫無錯誤的情況下編譯並執行，但作業所產生的結果暗藏錯誤，如果沒有仔細核對，不易發覺程式的結果有誤。

以下列程式碼為例，程式中使用 calcMPH()方法計算每小時平均速率（miles per hour），原本 1.6 小時移動距離 20.0 哩，MPH 應該為 12.5，但程式在調用 calcMPH()時，引數順序錯位了，使得程式執行結果得到 MPH 為 0.08，但程式碼在編譯及執行時並不會有錯誤訊息，只能靠程式設計師檢查分析進而修正錯誤。

```
private double calcMPH (double miles, double hours) {
    // TODO Auto-generated method stub
    return (miles/hours);
}
...
txtmsg = (TextView) findViewById(R.id.textView1);
double hours = 1.6;
double miles = 20.0;
double mph = calcMPH (hours, miles);
txtmsg.setText (String.valueOf(mph));
```

邏輯錯誤是相當棘手的問題，對於這類錯誤，除了靠程式設計者本身經驗外，還要善用除錯工具仔細追蹤、分析、測試，才能檢查出邏輯錯誤之所在並加以修正。例如設置中斷點、單步執行、檢查變數值...等，有關 Eclipse 除錯工具的使用將在下一節進一步介紹。

10.2 Eclipse 除錯選項

ADT 已將 Android SDK 所提供的基本的除錯工具 (包括 ADB、DDMS 及 Logcat)整合到 Eclipse 環境。

在 Eclipse 整合環境中可使用二種模式測試程式：一般模式（Run）及除錯模式（Debug）。我們可由功能表『Run/Debug』或直接按下工具列的 Debug 按鈕，選擇使用除錯模式來測試應用程式。

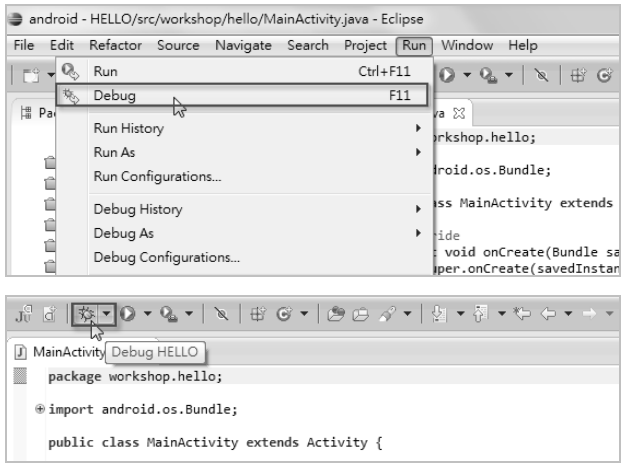

圖 10-2 使用除錯模式測試程式

10.2.1 顯示程式碼行號

在程式碼視窗之程式敘述前顯示行號可方便對照除錯訊息。如果目前程式碼視窗未顯示行號，可透過下列方式開啟行號顯示功能：

STEP 1 點選功能表『Window/Preferences』開啟「Preferences」對話視窗

STEP 2 由左邊面板點選『General/Editors/Text Editors』節點，在「Text Editors」的設定項目中勾選「Show line numbers」。

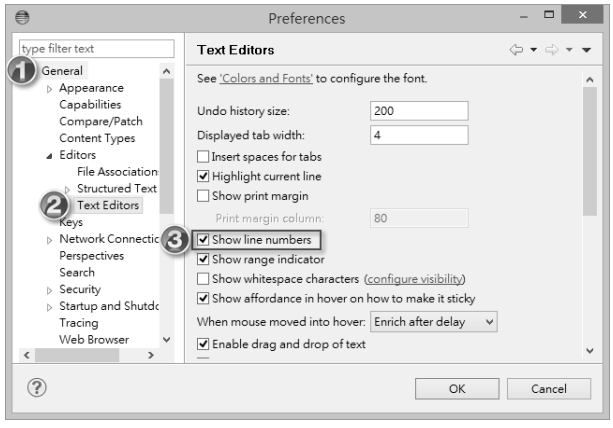

圖 10-3 由環境偏好設定開啓顯示行號功能

或由編輯區前端訊息區之快顯功能表，選擇「Show Line Numbers」。

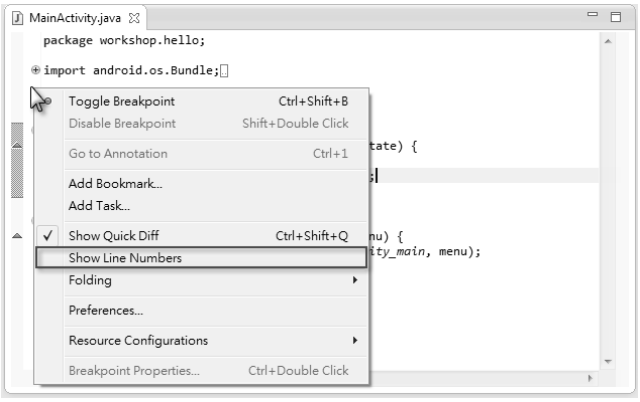

圖 10-4 由編輯區前端訊息區快顯功能表開啓顯示行號功能

10.2.2 設定中斷點

「中斷點」顧名思義就是用來暫停程式執行。在除錯式下，當程式執行到中斷點之敘述時就會暫停。通常我們會先研判程式可能發生問題的程式區塊，並在該程式區塊處設置中斷點，然後在程式中斷暫停後，透過單步執行、查看變數或運算式內容等方式來檢驗程式的執行狀態，驗證程式碼的每一敘述執行結果是否與我們預期結果相同。

我們可直接由編輯區前端訊息區或前端灰色區域之快顯功能表，選擇「Toggle Breakpoint」來新增中斷點或移除先前加入的中斷點，或按 Ctrl＋Shift＋B 組合鍵來設定/清除目前游標所在敘述之中斷點。被設置中斷點的敘述，會在程式面板前端灰色區域標示藍色實心圓點。

對於已設置的中斷點則可由快顯功能表命令「Disable Breakpoint」暫時停用（標示白色實心圓點），對於被暫停使用的中斷點則可由快顯功能表命令「Enable Breakpoint」重新啟用。

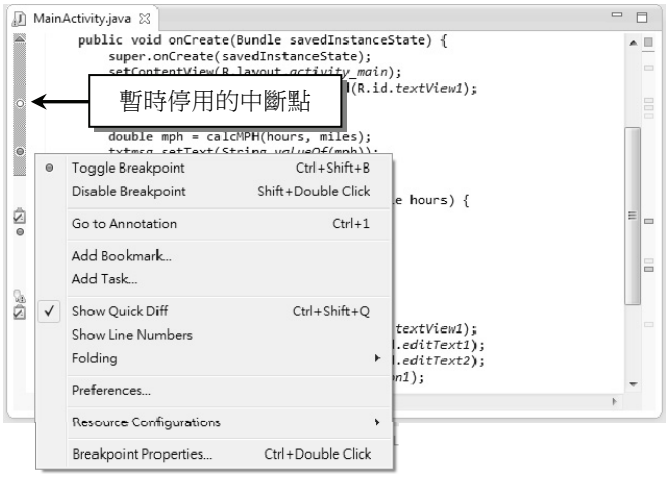

圖 10-5 中斷點之標示符號及快顯命令

由快顯功能表命令「Breakpoint Properties...」，可進一步對中斷點設定中斷條件，包括：

- Enabled　取消勾選表示暫停使用中斷點
- Hit count　執行幾次後才中斷。
- Conditional　指定的條件成立時才中斷。

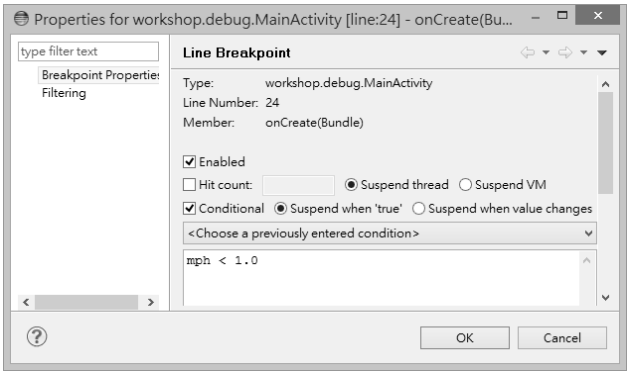

圖 10-6 設置條件式中斷

設置中斷點後，使用除錯模式執行程式，我們就可切換到除錯模式版面配置，進行進一步除錯作業。

10.2.3 追蹤（Trace）程式

在除錯模式版面配置下，可利用追蹤程式功能來驗證程式執行之細部流程，以檢驗流程是否異常。

- 繼續執行（Resume）
  由程式中斷處繼續執行程式，快速鍵 F8。
- 暫停執行（Suspend）
  中斷程式執行。
- 中止執行（Terminate）
  中止程式執行，快速鍵 Ctrl＋F2。
- 單步執行（Step Into）
  繼續執行一行敘述，如果是調用方法敘述，會進入方法程式碼追蹤細部流程，快速鍵 F5。
- 單程序執行（Step Over）
  繼續執行一行敘述，如果是調用方法敘述，會把整個方法程式碼視為單一敘述，一次執行完畢，快速鍵 F6。
- 執行完目前函式（Step Return）
  執行完目前方法所有敘述，返回調用方法的下一行敘述，快速鍵 F7。

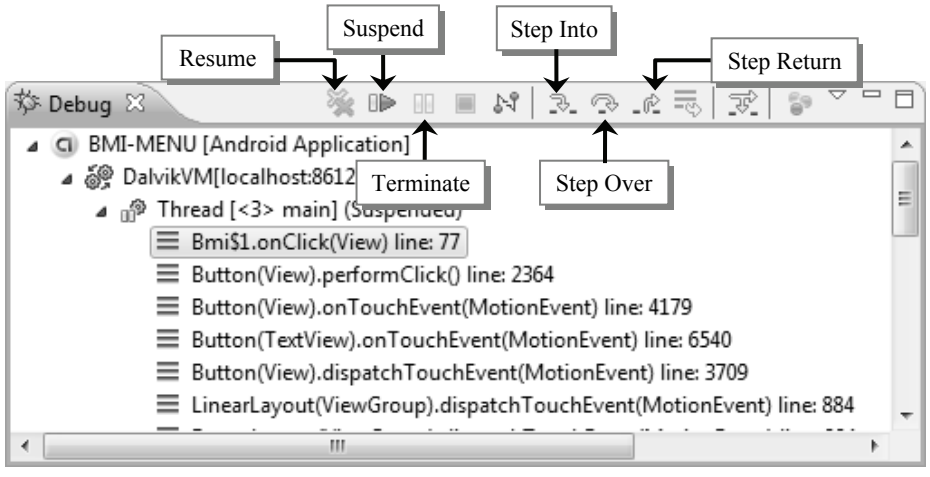

圖 10-7 中斷模式執行命令

10.2.4 監看變數值

在除錯模式下，Variables 窗格（由『Window/Show View/Variables』功能表命令來開啟）會顯示目前程序所使用的變數及其變數值。我們可從中驗證程式運行過程中，變數內容是否與預期結果相符。

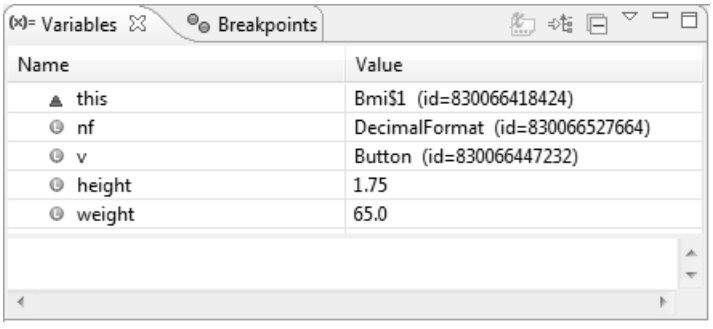

圖 10-8 中斷模式之變數檢視窗格

另外，也可透過 Expressions 窗格（由『Window/Show View/Expressions』功能表命令來開啟）監看運算式執行結果來檢查程式的執行狀態，以確認資料的正確性。

10.3 使用 try...catch

有時我們無法預期程式執行時會遇到什麼狀況，尤其關係到資料 I/O 作業時，可能遇到突發狀況，在這種情況下最好使用 try...catch...finally 來架構作業程式碼。

```
try…catch 程式架構如下：
    try {
        一般程式流程;
    } catch (Exception obj){
        錯誤處理流程;
    } finally {
        最後共通處理流程;
    }
```

套用 try...catch 架構時，當一般程式流程中有指令造成執行異常時，就會自動跳轉至對應的錯誤處理流程，通常會在錯誤處理流程中顯示錯誤原因。程式中可針對不同的異常原因提供不同的 catch 子句，而 finally 區塊則是通用的結束處理。

以下程式碼嘗試將 EditText 內的資料寫入到 SD 記憶卡中的 test.doc 檔，並使用 Toast 顯示 "寫入檔案成功" 訊息。

```
String SD_PATH = Environment.getExternalStorageDirectory().toString();
String FILE_PATH = "/sample/";
String INPUT_FILENAME = "test.doc";
private EditText mEditText;
...
try {
    File outFile = new File(SD_PATH + FILE_PATH + INPUT_FILENAME);
    FileOutputStream fileos = new FileOutputStream(outFile);
    mEditText = (EditText)findViewById(R.id.EditText01);
    fileos.write(mEditText.getText().toString().getBytes());//註[1]
    fileos.close();
    Toast.makeText(MainActivity.this,
        "寫入檔案成功", Toast.LENGTH_SHORT).show();
}
catch (FileNotFoundException e2) {
    e2.printStackTrace();
}
catch (IOException e) {
    e.printStackTrace();
}
```

10.4 檢視 Delvik Debug Monitor

Delvik debug monitor 是 Android SDK 提供的一項除錯工具，執行 Android SDK 的 tools 子目錄下 ddms.bat 即可啟動它。

---

[1] 如果沒有在應用程式的 AndroidManifest.xml 加入<uses-permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE" />權限宣告，就會造成執行異常。

在 DDMS 中可檢視 Android 裝置狀態，包括 CPU load、Memory usage、Proccess 狀態、應用程式狀態、Logcat 內容等，也可用來廣播狀態訊息、模擬電話呼叫、接收 SMS、虛擬地理座標等，此外功能表「Device/Screen capture」命令可截取裝置畫面；而點選功能表『Device/File Explorer...』則可開啟檔案總管，管理裝置之檔案。

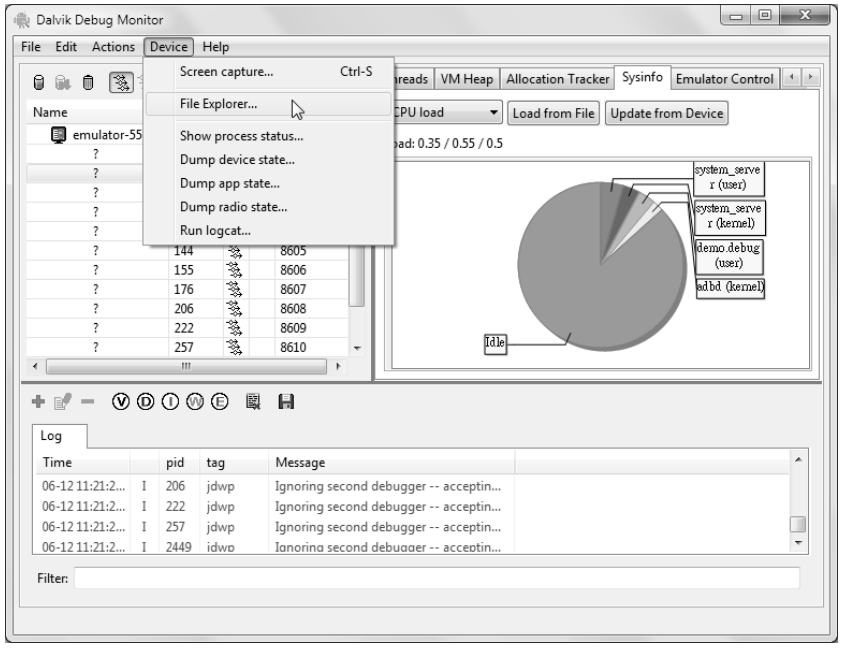

圖 10-9 Delvik debug monitor 執行畫面

DDMS 除了可獨立運行外，ADT 已整合 DDMS 到 Eclipse，我們可直接由 Eclipse『Window/Open Perspective』子功能表，選擇開啟 DDMS 版面配置（如圖 10-10 所示），切換到 DDMS 作業面板。其中「Devices」窗格工具列之「Screen capture」可截取裝置畫面；「File Explorer」窗格可管理裝置檔案內容，從 Android 裝置匯出檔案或將檔案匯入到 Android 裝置；「Emulator Control」窗格可模擬廣播狀態訊息、電話呼叫、接收 SMS、GPS 定位座標；「LogCat」窗格則可查看系統運行作業日誌，提供程式偵錯重要參考訊息。

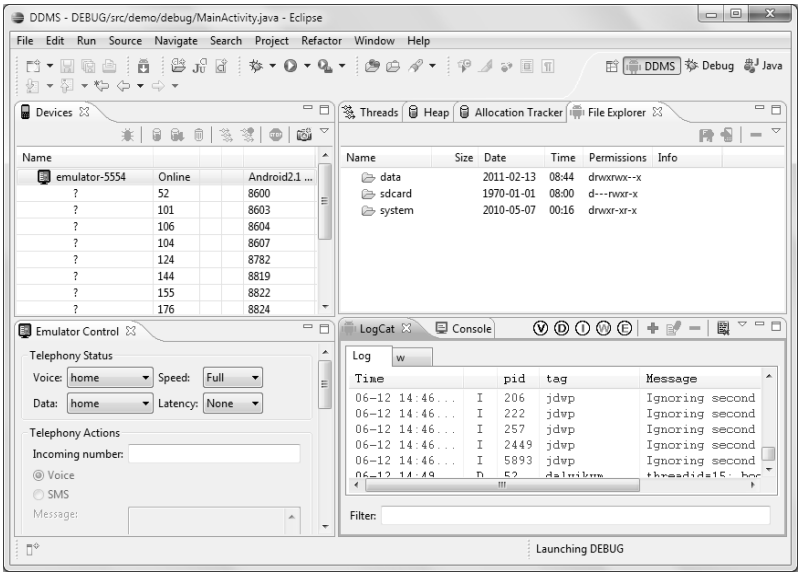

圖 10-10 Eclipse 之 DDMS 版面配置

10.5 使用 Snippets 管理程式碼

在開發 Apps 過程中，有些程式碼經常會重複使用，如果能將常用的程式碼整理分類，對後續開發應用程式非常有幫助。

10.5.1 建立 Snippets

例如使用 AlertDialog 建立訊息視窗，程序上皆需先建立 AlertDialog.Builder 物件，再透過 setTitle() 方法設定訊息視窗標題，setIcon() 方法設定訊息視窗圖示，使用 setMessage() 方法設定所要顯示的訊息內容，其程式碼基本架構如下：

```
AlertDialog.Builder dlg = new AlertDialog.Builder(MainActivity.this);
dlg.setTitle (getString(R.string.app_name));
dlg.setIcon(R.drawable.ic_launcher);
dlg.setMessage (getString(R.string.hello_world));
dlg.setPositiveButton ("YES", null);
```

如果想把上述程式碼加入程式碼片段庫，則只要選取這段程式碼，然後按下滑鼠右鍵，選擇「Add to Snippets...」[2]。

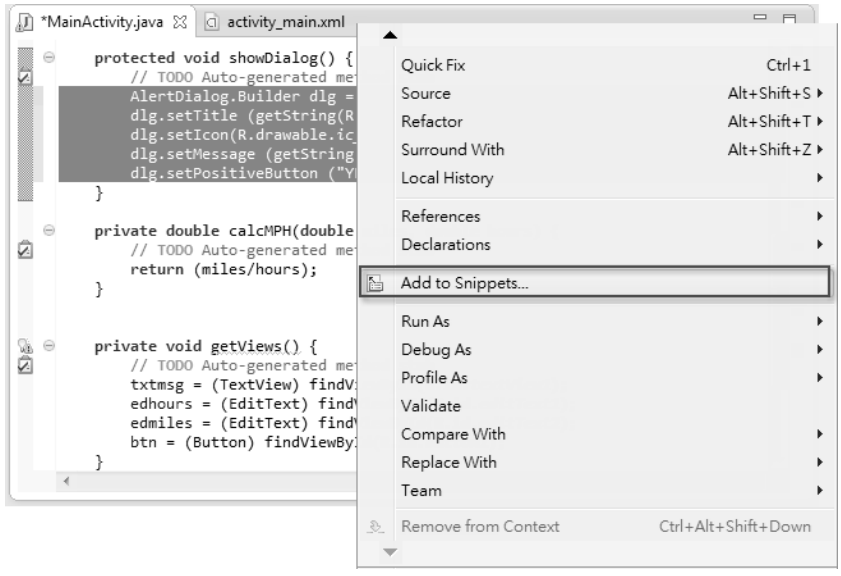

圖 10-11 新增程式碼片段到「Snippets」程式庫

如果是第一次使用 Snippets 功能，Eclipse 會要求輸入 Snippets 群組名稱；而如果不是第一次使用，則可由下接式清單選擇程式片段所要加入的群組。

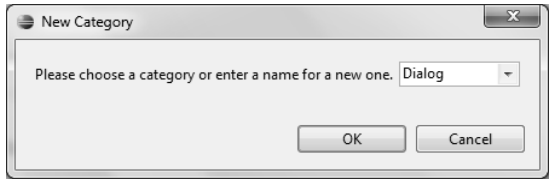

圖 10-12 建立「Snippets」群組名稱

[2] 如果是使用 Eclipse Juno 版本，預設並未加掛 Code Snippets，但只要安裝「Eclipse Java Web Developer Tools」就可使用。點選功能表『Help/Install New Software...』命令，選擇由 Juno 載點安裝更新，再由「Web, XML, Java EE and OSGi Enterprise Development」節點中勾選「Eclipse Java Web Developer Tools」。

之後 Eclipse 就會開啟「Customize Palette」對話視窗，讓我們將選取的程式碼新增到程式碼片段庫。

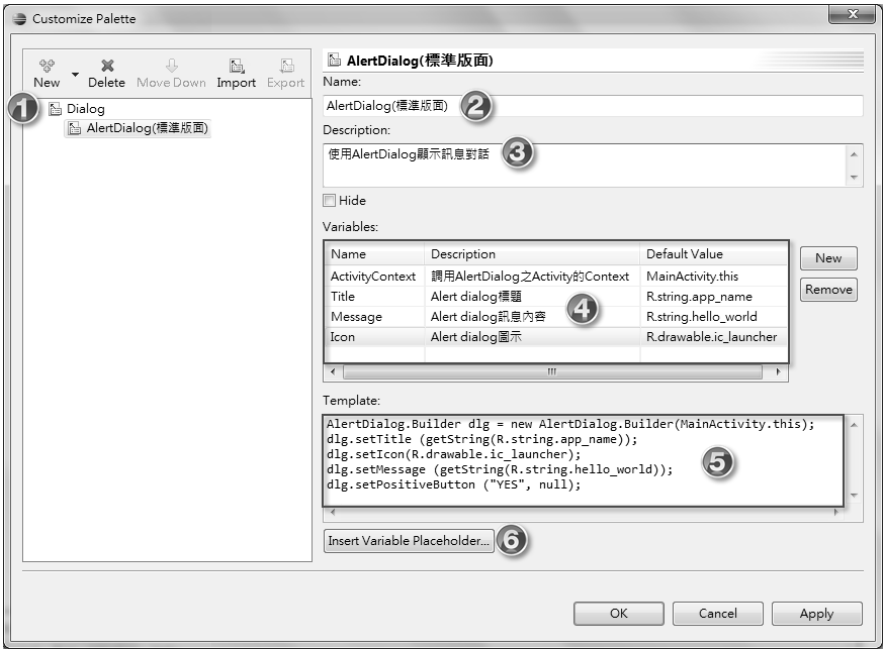

圖 10-13 新增「Snippets」作業畫面

❶ Snippets 程式群組名稱。

❷ 程式碼片段名稱，用來代表剛才選取加入 Snippets 的程式碼片段，例如：AlertDialog（標準版面）。

❸ 程式碼片段功能說明，例如：使用 AlertDialog 顯示訊息對話。

❹ 程式碼片段中所要使用的變數，預設是空的；可點擊右側 New 按鈕來加入變數宣告，在本例中我們加入了 ActivityContext、Title、Message 及 Icon 等 4 個變數宣告，這些變數可用來為程式碼片段提供參數置換功能（詳見❻）。

❺ 要加入 Snippets 的程式碼片段。

❻ 在程式碼片段中插入❹中所宣告的變數。由於程式碼片段會在不同的 Apps 中使用時，部份程式碼必須隨之修正，例如 AlertDialog 的標題、所要顯示的訊息等，都需要應用程式專案的需求而修正。

為使程式碼片段更具彈性使用，通常會將程式碼片段中需要修正的部份改以變數取代，例如剛才的程式碼片段如下：

```
AlertDialog.Builder dlg = new AlertDialog.Builder (MainActivity.this);
dlg.setTitle (getString(R.string.app_name));
dlg.setIcon (R.drawable.ic_launcher);
dlg.setMessage (getString(R.string.hello));
dlg.setPositiveButton ("YES", null);
```

程式碼中標示粗體底線的部份為在不同的應用程式專案套用 Snippet 時，必須修正的部份，因此我們這些程式碼分別置換為❹中所宣告的變數，其語法為${變數名稱}。在此直接按下「IInsert Variable Placeholder」，就可由已定義的變數清單中挑選變數並插入游標所在處，置換後的程式碼片段如下：

```
AlertDialog.Builder dlg = new AlertDialog.Builder(${ActivityContext});
dlg.setTitle (getString(${Title}));
dlg.setIcon(${Icon});
dlg.setMessage (getString(${Msg}));
dlg.setPositiveButton ("YES", null);
```

完成後，在 Eclipse 的 Snippets 窗格就可看到剛加入的 AlertDialog 程式碼片段。

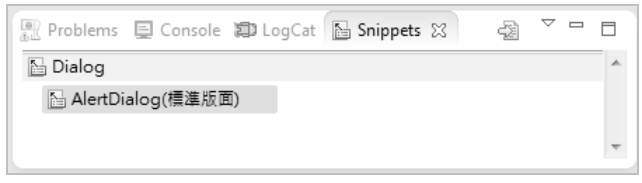

圖 10-14 「Snippets」窗格會列出目前程式碼片段庫內容

10.5.2 使用 Snippets

建立好的程式碼片段可方便地插入編輯中的程式碼。首先將游標移到要插入程式碼片段之位置，之後雙擊 Snippets 窗格中的程式碼片段名稱，即會顯示「Insert Template」對話視窗：

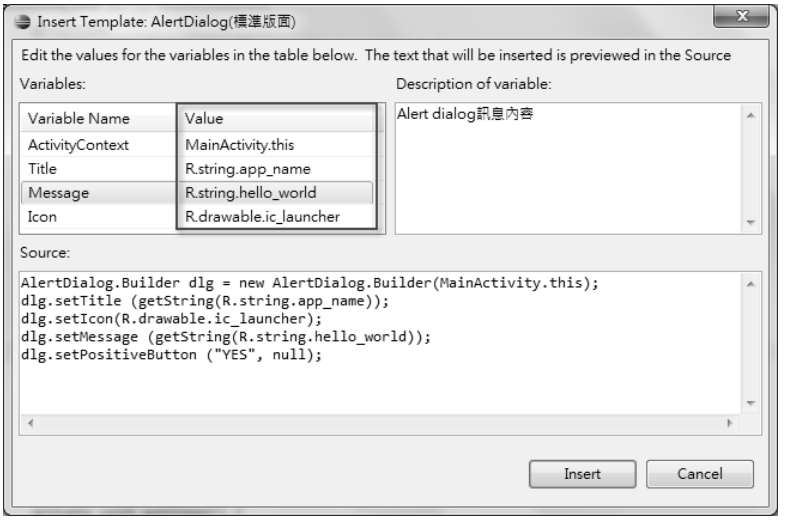

圖 10-15 插入「Snippets」作業畫面

如需要修改變數值，須直接在 Value 欄輸入新值即可，完成後按下「Insert」按鈕，程式碼片段就會自動插入到游標處。

除了程式碼外，Snippets 也可以用來管理應用程式的 XML 資料，用法都一樣。

實作練習10.1 程式除錯

本練習目的在熟練 Eclipse 工具修正程式錯誤。

STEP 1 匯入 DEBUG-LAB 專案到 Eclipse 工作區。

STEP 2 修正程式語法錯誤，編輯執行程式。

STEP 3 依據程式執行結果，修正程式邏輯錯誤。

STEP 4 確認程式執行結果正確性。

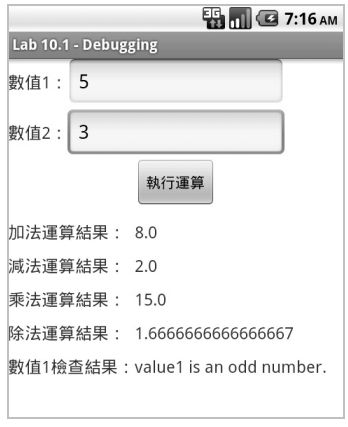

圖 10-16 修正 DEBUG-LAB 後的執行畫面