SA123

首頁

小技巧 | 在 Android Studio 除錯應用 （下）

2022年02月15日

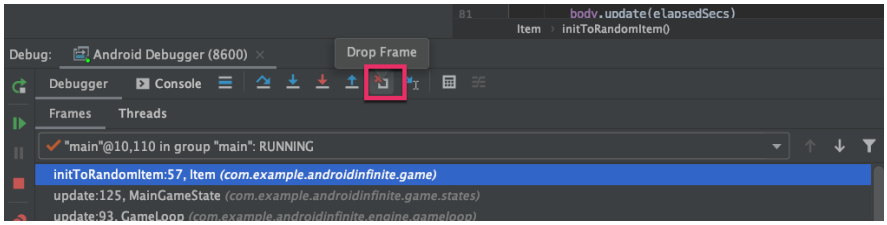
搜尋…

作為開發者，我們有時會被一些問題所困，導致在偵錯程式中所花費的時間甚至超過了編寫程式碼所用的時間。正因如此，最近我們找機會了解了 Android Studio 團隊在提升除錯速度方面使用的一些技巧。接下來，我們會為您一一呈現那些我們認為最好的、節省您時間的、且方便與您的除錯流程整合的小技巧。

雖然您的應用可能與本文假想中的示例應用大相徑庭，但是本文所介紹的小竅門可以用在任何應用的開發上。本文分為上下兩篇，上篇已於前段時間釋出，如果您還沒來得及閱讀上篇，請在這裡檢視《**在 Android Studio 中除錯您的應用 (上)**》。

Drop frame (丟棄當前幀)

有些時候，當您瀏覽掛起的程式碼時，可能會意外跳過某個本應該進入的方法。如果您的裝置執行的是 Android 10 或者更高版本，您可以透過點選除錯工具欄中的**Drop Frame**按鈕來進行回溯:



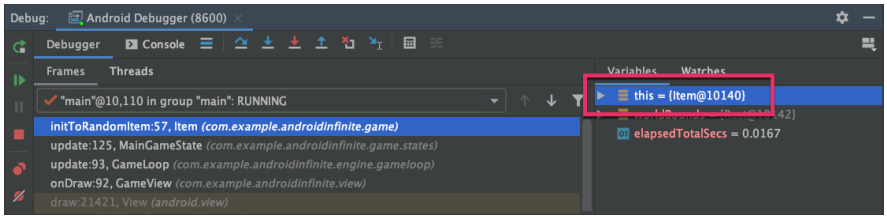
這個功能會把您從當前的方法帶回到其開始執行前的節點，從而給您一個重新進入該方法的機會。

此功能並不是 “時間機器” 。如果您正處於一個長函式的中間位置，而它此前已經執行了許多工作 (例如，修改了當

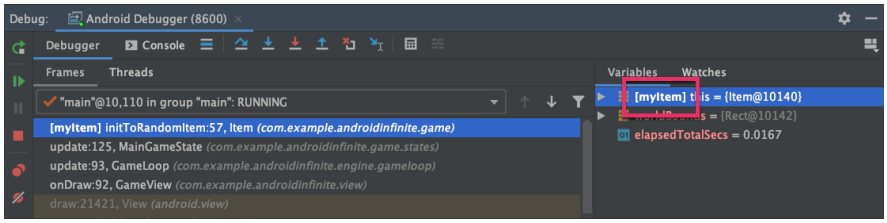
前類的狀態)。在您丟棄當前幀時，這類操作所產生的改變不會被撤銷。

Mark object (標記物件)

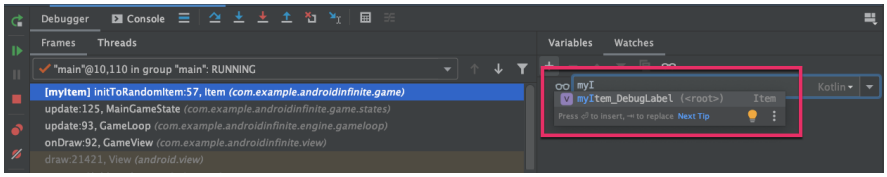
有時候，您會想要跟蹤某些特定型別例項的生命週期。本例中，要跟蹤的物件有一個雜湊值: @10140:



為了能夠在這個物件再次出現時可以認出它來，您可能已經掏出紙筆準備記下這個數字了。不過您也可以選擇另一種方式: 右擊該物件，點選**Mark Object**為其新增標籤。這樣一來，無論被標記的物件出現在除錯視窗的任何地方，它都會帶有您新增的標籤以方便辨認。這裡我們為該物件新增一個 “**myItem**” 標籤:

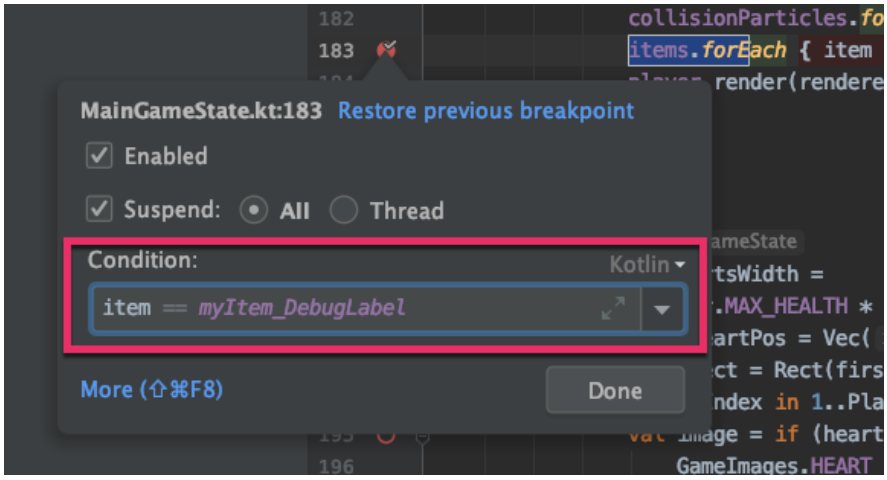


更棒的是，就算您處於完全不同的上下文，無法觸及到剛才的物件，您也可以在**Watches**視窗對其進行檢視。無論您處在什麼位置，只要觸發斷點，就可以在**Watches**視窗新增字尾為 “**_DebugLabel**” 的標籤 (不用擔心自己會不記得字尾的內容，這裡有自動補全):

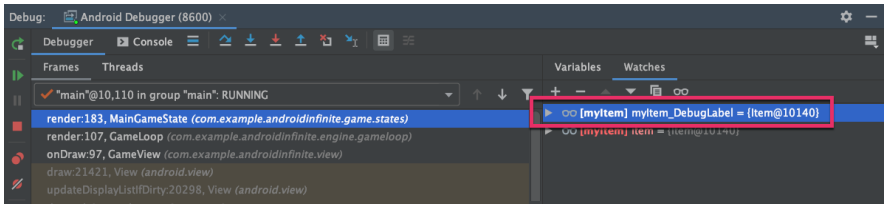


現在，您可以在任何地方使用**Watches**視窗來觀察該型別物件的狀態。

您也可以將此功能與條件斷點組合。舉例來說，您可打一個斷點，右擊併為其設定一個條件來檢查打了標籤的物件:

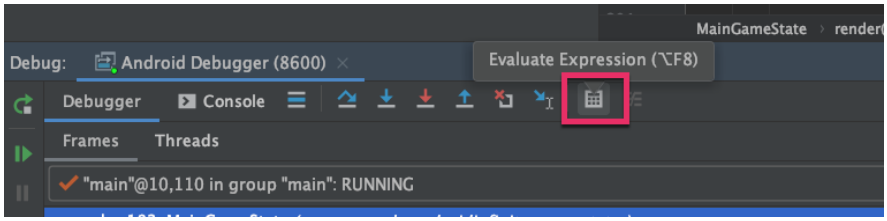


這樣一來，就不用在進入包含特定例項的範圍之前跳過一堆斷點，程式碼會執行到合適的地方再停止:

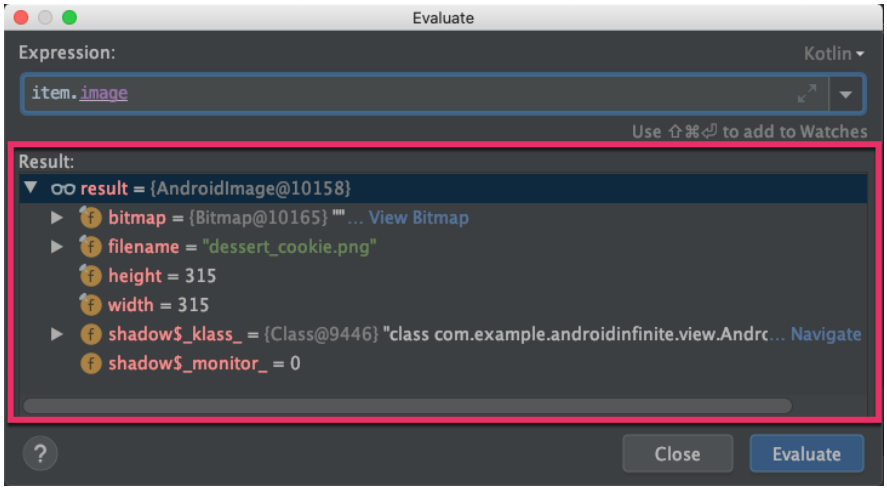


Evaluate expression (評估表示式)

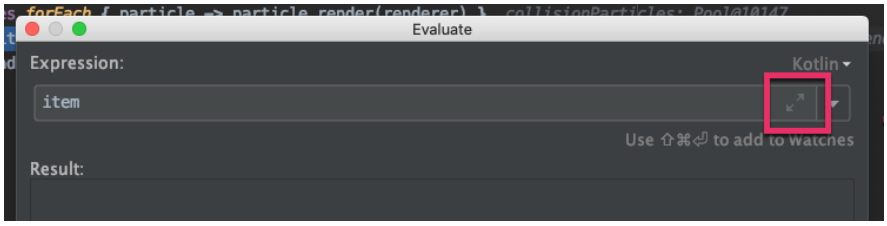
儘管**Variables**和**Watches**視窗對於跟蹤某個顯式值時已經十分好用，但您有時候還是會想要更加自由地探索您的程式碼，這時候就輪到評估表示式功能登場了。當您正處於某個斷點時，您可以使用除錯工具欄中的**Evaluate expression**按鈕來訪問這一功能。



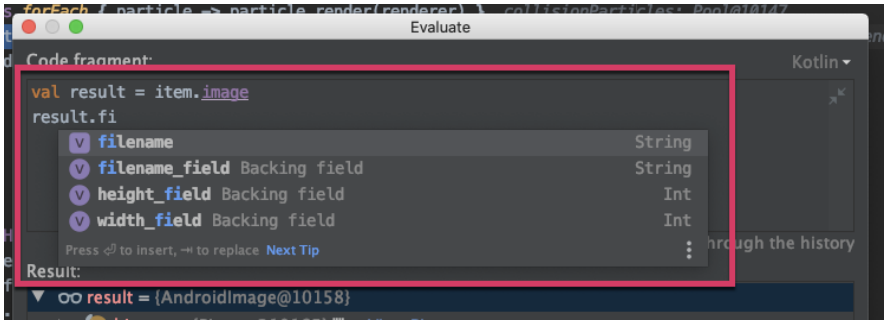
您可以在**Expression**輸入框中輸入任何表示式，點選**Evaluate**按鈕就可以對其進行評估。當然，如果您評估了一個物件，在評估完成後，您就可以在**Result**部分瀏覽該物件的詳細資訊:



評估表示式彈窗可能會以單行模式開啟，您可以透過點選**Expand**來將其擴充套件為多行模式:

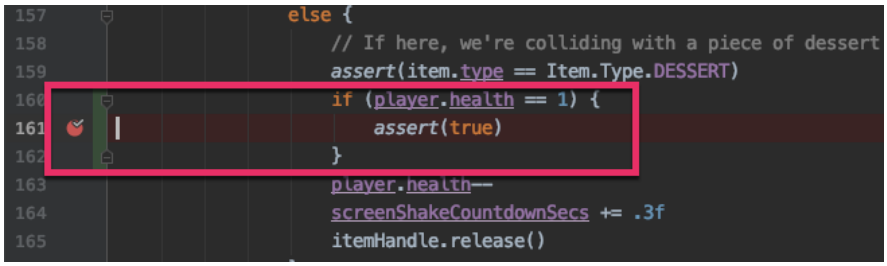


現在，您可以輸入複雜的多行表示式，其中可以包含變數、if 語句等各種內容:

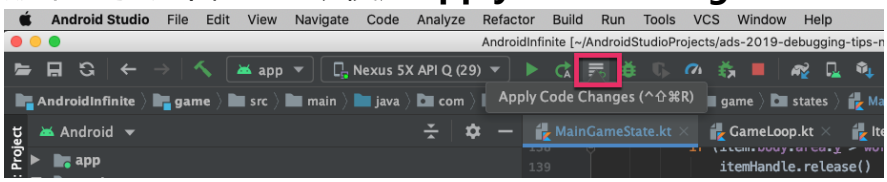


Apply changes

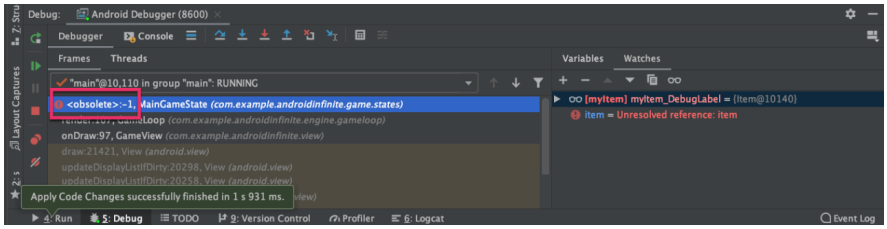
前面講過，當您使用條件斷點時，會需要評估一個表示式；即便程式碼沒有在斷點停止，偵錯程式依然需要執行評估操作。如果您在一個非常緊密的迴圈中執行評估操作，例如遊戲中的動畫處理，則可能導致應用停頓。儘管條件斷點很有用，但在某些情況下您可能無法依靠它們。解決此問題的一種方法是將條件表示式新增到程式碼中，並使用無操作 (no-op) 表示式，從而使其可以附加斷點:



在修改完程式碼之後，您可能會決定重啟應用並點選**Debug**按鈕，但是如果您的應用執行在 Android 8 或更高版本的系統中，您可以使用**Apply Code Changes**:

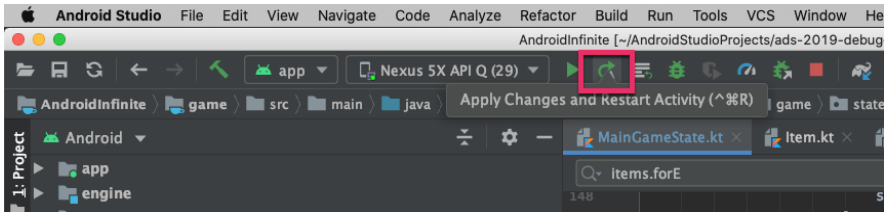


現在，嵌入的表示式已經修補了您的程式碼，但您仍然會在**Frames**視窗中看到您已經更新的方法被標記為**過時(Obsolete)**:



這是因為，雖然新的程式碼已經打好了補丁，但是偵錯程式指向的仍是舊的程式碼。您可以使用丟棄當前幀的功能來離開舊的方法，然後進入修改過的方法之中。

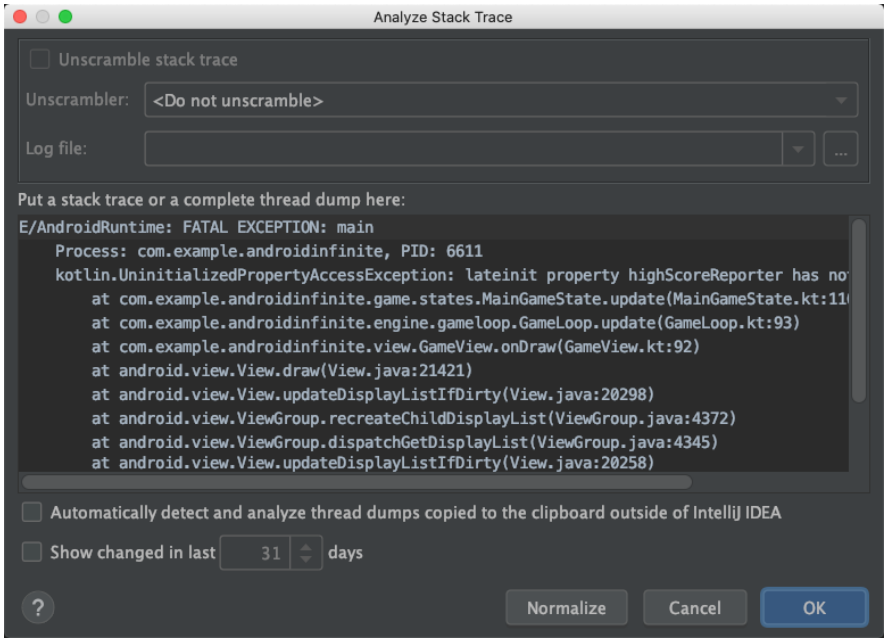
儘管本例中並不需要，但這裡還有一種選擇:**Apply Changes and Restart Activity**。不同於**Apply Code Changes**，該操作會重啟 Activity。如果您修改了佈局資源或您要除錯的程式碼，例如在 onCreate 方法中，該選項將非常有用。



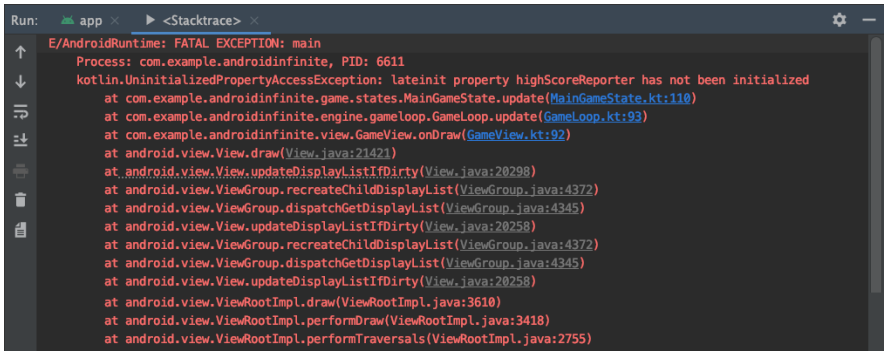
分析堆疊資訊

就算您掌握了所有的竅門與技巧，您的程式碼仍然可能出現 Bug，而您也會因此收到一些崩潰報告，這些報告中則可能包含了異常堆疊資訊的文字副本。您可以使用**Analyze**選單中的**Analyze Stack Trace or Thread Dump**將這些資訊轉化為有意義的內容。

此工具提供了一個貼上堆疊資訊的文字框，不過它也會自動填充系統剪貼簿中的文字:



點選**OK**之後，就會將包含完整註釋的堆疊資訊新增到控制檯:



您可以一眼看出來自您自己程式碼檔案的內容 (以藍色突出顯示) 與您可能不需要關注的程式碼 (以灰色突出顯示)。並且，您可以透過單擊連結在您的程式碼檔案中進行跳轉。

結語

本文提供了一些可以加快除錯速度的技巧和竅門。由於篇幅所限，更多技巧簡單歸納如下:

- 在 Debug 模式下，點選程式碼的行數數字可以直接執行此行程式碼
- Ctrl + 拖動操作可以複製斷點
- 您可以在函式的右括號處設定斷點
- 您可以在欄位和屬性上設定斷點，被稱為 “field watchpoints”
- 您可以為介面中的方法設定斷點，從而使它的所有實現都會觸發斷點

您可以透過下面這個影片再次回顧更多本文的細節和演示內容:

- 騰訊影片連結

  https://v.qq.com/x/page/o3030ha9b5e.html

- Bilibili 影片連結

  https://www.bilibili.com/video/av78114615/

也請您查閱更多與本話題相關的資源:

- Android Developer 官方文件 | 除錯預構建的 APK

  https://developer.android.google.cn/studio/debug/apk-debugger

- 透過資料瀏覽來控制資料在偵錯程式中的顯示方式

  https://www.jetbrains.com/help/idea/debugger-data-type-renderers.html

- 如何使用和理解 Overhead 選項卡

  https://www.jetbrains.com/help/idea/monitor-debugger-overhead.html

- Android Developer 官方文件 | Android Studio — 除錯您的應用

  https://developer.android.google.cn/studio/debug

- IntelliJ IDEA 除錯程式碼

  https://www.jetbrains.com/help/idea/debugging-code.html

推薦閱讀

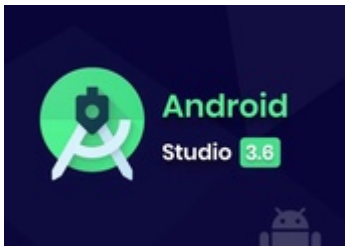
點選屏末**|閱讀原文|Android Developer 官方文件 | Android Studio — 除錯您的應用**

长按识别二维码

关注 谷歌开发者

谷歌中国官方账号 预测未来，不如创造未来 Google开发者

相關文章

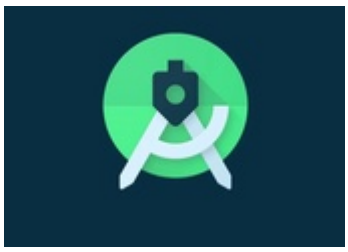


Android Studio 3.6 穩定版釋出

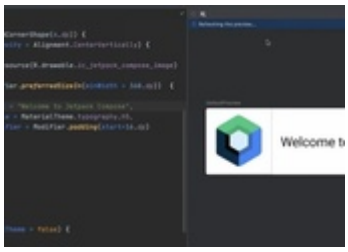
Android 11 正式釋出 | 開發者們的舞臺已就緒

Android Studio 4.1 釋出，全方位提升開發體驗

Android Studio 4.0 正式版釋出



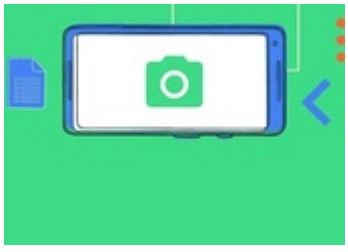
Android 11 新工具 | 提升應用私密性和穩定性



Jetpack Compose Alpha 版現已釋出！

聚焦 Android 11：Android 開發者工具

在 Android 11 及更高版本系統中處理可空性



Android Jetpack CameraX 庫 Beta 版正式釋出！

Android 樣式系統 | 主題背景屬性