链滴

Android Studio 常用快捷键

作者：yman0917
原文链接：https://ld246.com/article/1478248548394
来源网站：链滴

常用技巧

1. 书签（Bookmarks）

- **描述：**这是一个很有用的功能，让你可以在某处做个标记（书签），方便后面再跳转到此处。
- **调用：**Menu → Navigate → Bookmarks
- **快捷键：**
-
  - 添加/移除书签：F3(OS X) 、F11(Windows/Linux);
  - 添加/移除书签(带标记)：Alt + F3(OS X)、Ctrl + F11(Windows/Linux);
  - 显示全部书签：Cmd + F3(OS X) 、Shift + F11(Windows/Linux)，显示所有的书签列表，并是可以搜索的。
  - 上一个/下一个书签：无，可以在设置中设置快捷键。
- **更多：**当你为某个书签指定了标记，你可以使用快捷键 Ctrl + 标记 来快速跳转到标记处，比输入Ctrl + 1，跳到标记为1的书签处。

2. 折叠/展开代码块（Collapse Expand Code Block）

- **描述：**该操作提供一种方法，让你隐藏你不关心的部分代码，以一种较为简洁的格式显示关键码。一个有意思的用法是隐藏匿名内部类的代码，让其看起来像一个Lambda表达式。
- **快捷键：**Cmd + “+”/”-“(OS X)、Ctrl + Shift + “+”/”-“(Windows/Linux);
- **更多：**可以在Settig → Editor → General → Code Folding 中设置折叠规则。

3. 与分支比对（Compare With Branch (Git)）

- **描述：**假如你的项目是使用git来管理的，你可以将当前文件或者文件夹与其他的分支进行比对比较有用的是可以让你了解到你与主分支有多少差别。
- **调用：**Menu → VCS → Git → Compare With Branch

4. 与剪切板比对（Compare With Clipboard）

- **描述：**将当前选中的部分与剪切板上的内容进行比对。
- **调用：**右键选中的部分，在右键菜单中选择“Compare With Clipboard”。

5. 上下文信息（Context Info）

- **描述：**当前作用域定义超过滚动区域，执行该操作将显示所在的上下文信息，通常它显示的是名或者内部类类名或者当前所在的方法名。该操作在xml文件中同样适用。
- **调用：**Menu → View → Context Info
- **快捷键：**Alt + Q (Windows/Linux)
- **更多：**个人认为，这个功能更好的用法是快速查看当前类继承的父类或者实现的接口。

6. 查找操作（Find Action）

- **描述：**输入某个操作的名称，快速查找，对于没有快捷键的部分操作这是一个很有用的技巧。

- **快捷键：**Cmd +Shift + A(OS X)、Ctrl + Shift + A(Windows/Linux)；
- **更多：**当某个操作是有快捷键的，会显示在旁边。

7. 查找补全（Find Complection）

- **描述：**当你在一个文件中进行查找时，使用自动补全快捷键可以给出在当前文件中出现的建议词；
- **快捷键：**Cmd + F(OS X),Ctrl + F(Windows/Linux),输入一些字符，然后使用自动补全；

8. 隐藏所有面板（Hide All Panels）

- **描述：**切换编辑器铺满整个程序界面，隐藏其他的面板。再次执行该操作，将会回到隐藏前的态。
- **调用：**Menu → Window → Active Tool Window → Hide All Windows；
- **快捷键：**Cmd +Shift + F12(OS X)、Ctrl + Shift + F12(Windows/Linux)；

9. 高亮一切（Hightlight All the Things）

- **描述：**该操作将会高亮某个字符在当前文件中所有出现的地方。这不仅仅是简单的匹配，实际它会分析当前的作用域，只高亮相关的部分。
- **调用：**Menu → Edit → Find → Highlight Usages in File；
- **定位到上一处/下一处：**Menu → Edit → Find → Find Next/Previous；
- **快捷键：**相关快捷键请在菜单中查看；
- **更多：**
-
  - 如果高亮一个方法的 return或throw语句，将会高亮这个方法的所有出口/结束点；
  - 如果高亮某个类定义处的 extend或implements语句，将会高亮继承的或实现的方法；
  - 高亮一个 import语句也会高亮使用到的地方；
  - 按下Esc可以退出高亮模式；

10. 回到上一个工具窗口（Jump to Last Tool Window）

- **描述：**有时候你会从某个工具窗口跳到编辑器里面，然后又需要重新回到刚才操作的那个工具，比如你查找使用情况的时，使用该操作可以在不使用鼠标的情况下跳转到之前的工具窗口。
- **快捷键：**F12；

11. 上一个编辑位置（Last Edit Location）

- **描述：**该操作将使得你导航到上一处你改动过的地方，这与点击工具栏上的返回箭头回到上一定位位置是不一样的，该操作将会返回到上一个编辑的位置。
- **快捷键：** Cmd + Shift + Delete(OS X)、Ctrl + Shift + Backspace(Windows/Linux);

12. 在方法和内部类之间跳转（Move Between Methods and Inner Classes）

- **描述：**该操作让光标在当前文件的方法或内部类的名字间跳转。
- **调用：**Navigate → Next Method/Previous Method;

- **快捷键：**Ctrl + Up/Down(OS X)、Alt + Up/Down(Windows/Linux);

13. 定位到嵌套文件（Navigate to Nested File）

- **描述：**有时你有一堆存放在不同目录下的同名文件，例如不同模块下的 AndroidManifest.xml 件，当你想定位到其中的一个文件，你会得到一堆搜索结果，你还得辨认哪个才是你需要的。通过在索框中输入部分路径的前缀，并添加斜杠号，你就可以在第一次尝试的时候就找到正确的那个。
- **快捷键：**Shift + Cmd + O(OS X)、Shift + Ctrl + N(Windows/Linux);

14. 定位到父类（Navigate to parent）

- **描述：**如果光标是在一个继承父类重写的方法里，这个操作将定位到父类实现的地方。如果光是在类名上，则定位到父类类名。
- Menu → Navigate → Super Class/Method
- **快捷键：**Cmd + U(OS X)、Ctrl + U(Windows/Linux);

15. 根据编号打开面板（Open a Panel by Its Number）

- **描述：**你可能已经注意到某些面板的名称左边有一个数字，这里有个快捷操作可以打开它们。果你没看到面板的名称，请点击IDE的左下角的切换按钮。
- **快捷键：**Cmd + 数字(OS X)、Alt + 数字(Windows/Linux);

16. 在外部打开文件（Open File Externally）

- **描述：**通过这个快捷键，简单地点击Tab，就可以打开当前文件所在的位置或者该文件的任意层路径。
- **快捷键：**Cmd + 单击Tab(OS X)、Ctrl + 点击Tab(Windows/Linux);

17. 参数信息（Parameter Info）

- **描述：**这个操作将显示和你在方法声明处写一样的参数列表，当你想看某个存在的方法的参数这是一个很有用的操作。光标下的参数显示为黄色，如果没有参数显示黄色，意味着你的方法调用是效的，很可能是某个参数分配不对。（例如一个浮点数赋值给了整型参数）。如果你正在写一个方法用，突然离开编辑的地方，再返回的时候，输入一个逗号，就可以重新触发参数信息。
- **快捷键：**Cmd + P(OS X)、Ctrl + U(Windows/Linux);

18. 快速查看定义（Quick Definition Lookup）

- **描述：**你曾经是否想查看一个方法或者类的具体实现，但是不想离开当前界面？ 该操作可以帮搞定。
- **快捷键：**Alt + Space / Cmd + Y(OS X)、Ctrl + Shift + I(Windows/Linux)

19. 最近修改的文件（Recently Changed Files）

- **描述：**该操作类似于“最近访问（Recents）”弹窗，会显示最近本地修改过的文件列表，根修改时间排列。可以输入字符来过滤列表结果。
- **快捷键：**Cmd + Shift + E(OS X)、Ctrl + Shift + E(Windows/Linux)

20. 最近访问（Recents）

- **描述：**该操作可以得到一个最近访问文件的可搜索的列表。
- **快捷键：**Cmd + E(OS X)、Ctrl + E(Windows/Linux)

21. 相关文件（Related File）

- **描述：**该操作有助于在布局文件和Activity/Fragment之间轻松跳转。这也是一个快捷操作，在名/布局顶端的左侧。
- **快捷键：**Ctrl + Cmd + Up(OS X)、Ctrl + Alt + Home(Windows/Linux)

22. 返回到编辑器（Return to the Editor）

- **描述：**一大堆快捷键操作会把你从编辑器带走（type hierarchy, find usages, 等等）。如果你返回到编辑器，你有两个选项：
-
    1. Esc：该操作仅仅把光标移回编辑器。
    2. Shift + Esc：该操作会关闭当前面板，然后把光标移回到编辑器。
- **快捷键：**
-
    - 返回但保留打开的面板：Esc
    - 关闭面板并返回：Shift + Esc

23. Select In

- **描述：**拿着当前文件然后问你在哪里选中该文件。恕我直言，最有用的就是在项目结构或者资管理器中打开该文件。每一个操作都有数字或者字母作为前缀，可以通过这个前缀来快速跳转。通常我会 Alt + F1 然后 回车(Enter) 来打开项目视图，然后 再用 Alt + F1 在OS X的Finder里找到文件。可以在文件中或者直接在项目视图里使用该操作。
- **快捷键：**Alt + F1；

24. 扩大/缩小选择（Extend/Shrink Selection）

- **描述：**该操作会在上下文逐渐扩大/缩小当前选择范围。例如，它会先选中当前变量，再选中当语句，然后选中整个方法，缩小选择则相反。
- **快捷键：**Alt + 上/下 (OS X)、Ctrl+W / Ctrl + Shift + W（Windows、Linux）

25. Sublime Text式的多处选择（Sublime Text Multi Selection）

- **描述：**这个功能超级赞！该操作会识别当前选中字符串，选择下一个同样的字符串，并且添加个光标。这意味着你可以在同一个文件里拥有多个光标，你可以同时在所有光标处输入任何东西。
- **快捷键：**Ctrl + G(OS X)、Alt + J （Windows、Linux）

26. 文件结构弹窗（The File Structure Popup）

- **描述：**该操作可以展示当前类的大纲，并且可以快速跳转。你还可以通过键盘输入来过滤结果这是一种很高效的方法来跳转到指定方法。
- **更多：**

-
  - 你在输入字符的时候可以用驼峰风格来过滤选项。比如输入”oCr”会找到”onCreate”
  - 你可以通过勾选多选框来决定是否显示匿名类。这在某些情况下很有用，比如你想直接跳转到个OnClickListener的onClick方法。
- **快捷键：**Cmd + F12(OS X)、Ctrl + F12(Windows/Linux)
- **调用：**Menu → Navigate → File Structure

27. 切换器（The Switcher）

- **描述：**该快捷键基本上就是IDE的alt+tab/cmd+tab命令。你可以用它在导航tab或者面板切换一旦打开这个窗口，只要一直按着ctrl键，你可以通过对应的数字或者字母快捷键快速选择。你也可通过backspace键来关闭一个已选中的tab或者面板。
- **快捷键：**Ctrl + Tab

28. 版本控制操作弹窗（VCS Operations Popup）

- **描述：**该操作会给你显示最常用的版本控制操作。如果你的项目没有用git等版本控制软件进行理，它至少会给你提供一个由IDE维护的本地历史记录。
- **快捷键：**Ctrl + V(OS X)、Alt + `(Windows/Linux)

编码技巧

29. 列选择/块选择（Column Selection）

- **描述：**正常选择时，当你向下选择时，会直接将当前行到行尾都选中，而块选择模式下，则是据鼠标选中的矩形区域来选择。
- **调用：**按住Alt，然后拖动鼠标选择。
- 开启/关闭块选择：Menu → Edit → Column Selection Mode
- **快捷键：**切换块选择模式：Cmd + Shift + 8(OS X)、Shift + Alt + Insert(Windows/Linux);

30. 语句补全（Complete Statement）

- **描述：**这个方法将会生成缺失的代码来补全语句，常用的使用场景如下：
-
  - 在行末添加一个分号，即使光标不在行末；
  - 为if、while、for 语句生成圆括号和大括号；
  - 方法声明后，添加大括号；
- **调用：**Menu → Edit → Compelete Current Statement
- **快捷键：**Cmd + Shift + Enter(OS X)、Ctrl + Shift + Enter(Windows/Linux)；
- **更多：**如果一个语句已经补全，当你执行该操作时，则会直接跳到下一行，即使光标不在当前的行末。

31. 删除行（Delete Line）

- **描述：**如果没选中，则删除光标所在行，如果选中，则会删除选中所在的所有行。
- **快捷键：**Cmd + Delete(OS X)、Ctrl + Y(Windows/Linux)

32. 行复制（Duplicate Line）

- **描述：**复制当前行，并粘贴到下一行，这个操作不会影响剪贴板的内容。这个命令配合移动行捷键非常有用。
- **快捷键：**Cmd + D(OS X)、Ctrl + D(Windows/Linux)

33. 编写正则表达式（Edit Regex）

- **描述：**使用Java编写正则表达式是一件很困难的事，主要原因是：
-
  - 你必须得避开反斜杠；
  - 说实话，正则很难；
  - 看第二条。

IDE能帮我们干点啥呢？当然是一个舒服的界面来编写和测试正则啦~ - **快捷键：**Alt + Enter → c eck regexp

34. 使用Enter和Tab进行代码补全的差别（Enter vs Tab for Code Completion）

- **描述：**代码补全时，可以使用Enter或Tab来进行补全操作，但是两者是有差别的。
- 使用Enter时：从光标处插入补全的代码，对原来的代码不做任何操作。
- 使用Tab时：从光标处插入补全的代码，并删除后面的代码，直到遇到点号、圆括号、分号或空格止。

35. 提取方法（Extract Method）

- **描述：**提取一段代码块，生成一个新的方法。当你发现某个方法里面过于复杂，需要将某一段码提取成单独的方法时，该技巧是很有用的。
- **调用：**Menu → Refactor → Extract → Method
- **快捷键：**Cmd + Alt + M(OS X)、Ctrl + Alt + M(Windows/Linux)；
- **更多：**在提取代码的对话框，你可以更改方法的修饰符和参数的变量名。

36. 提取参数（Extract Parameter）

- **描述：**这是一个提取参数的快捷操作。当你觉得可以通过提取参数来优化某个方法的时候，这技巧将很有用。该操作会将当前值作为一个方法的参数，将旧的值放到方法调用的地方，作为传进来参数。
- **调用：**Menu → Refactor → Extract → Parameter
- **快捷键：**Cmd + Alt + P(OS X)、Ctrl + Alt + P(Windows/Linux)；
- **更多：**通过勾选“delegate”，可以保持旧的方法，重载生成一个新方法。

37. 提取变量（Extract Variable）

- **描述：**这是一个提取变量的快捷操作。当你在没有写变量声明的直接写下值的时候，这是一个

方便生成变量声明的操作，同时还会给出一个建议的变量命名。

- **调用：**Menu → Refactor → Extract → Variable
- **快捷键：**Cmd + Alt + V(OS X)、Ctrl + Alt + V(Windows/Linux)；
- **更多：**当你需要改变变量声明的类型，例如使用 List 替代 ArrayList，可以按下Shift + Tab，会显示所有可用的变量类型。

38. 内置（Inline）

- **描述：**当你开始对提取操作有点兴奋的时候，突然觉得东西太多了，怎么办呢？这是一个和提相反的操作。该操作对方法、字段、参数和变量均有效。
- **调用：**Menu → Refactor → Inline
- **快捷键：**Cmd + Alt + N(OS X)、Ctrl + Alt + N(Windows/Linux)；

39. 合并行和文本（Join Lines and Literals）

- **描述：**这个操作比起在行末使劲按删除键爽多了！该操作遵守格式化规则，同时：
-
  - 合并两行注释，同时移除多余的 //；
  - 合并多行字符串，移除 +和双引号；
  - 合并字段的声明和初始化赋值；
- **快捷键：**Ctrl + Shift + J；

40. 动态模板（Live Templates）

- **描述：**动态模板是一种快速插入代码片段的方法，使用动态模板比较有意思的是你可以使用合的默认值将模板参数化，当你插入代码片段时，这可以指导你完成参数。
- **更多：**如果你知道模板的缩写，就可以不必使用快捷键，只需要键入缩写并使用Tab键补全即。
- **快捷键：**Cmd + J(OS X)、Ctrl + J(Windows/Linux);

41. 上下移动行（Move Lines Up Down）

- **描述：**不需要复制粘贴就可以上下移动行了。
- **快捷键：**Alt + Shift + Up/Down；

42. 移动方法（Move Methods）

- **描述：**这个操作和移动行操作很类似，不过该操作是应用于整个方法的，在不需要复制、粘贴情况下，就可以将整个方法块移动到另一个方法的前面或后面。该操作的实际叫做“移动语句”，这味着你可以移动任何类型的语句，你可以方便地调整字段或内部类的顺序。
- **快捷键：**Cmd + Alt + Up/Down(OS X)、Ctrl + Shift + Up/Down(Windows/Linux);

43. 取反补全（Negation Completion）

- **描述：**有时你自动补全一个布尔值，然后回到该值的前面添加一个感叹号来完成取反操作，现通过使用输入!代替enter完成补全操作，就可以跳过这些繁琐的操作了。

- **快捷键：**代码补全的时候，按下 !即可（有时需要上下键选中候选项）；

44. 后缀补全（Postfix Completion）

- **描述：**你可以认为该操作是一种代码补全，它会在点号之前生成代码，而不是在点号之后。实上你调用这个操作和正常的代码补全操作一样：在一个表达式之后输入点号。

例如对一个列表进行遍历，你可以输入myList.for，然后按下Tab键，就会自动生成for循环代码。

- **调用：** 你可以在某个表达式后面输入点号，出现一个候选列表，在常规的代码补全提示就可以看到系列后缀补全关键字，同样的，你也可以在Editor → Postfix Completion中看到一系列后缀补全关字。
- 常用的有后缀补全关键字有：
-
  - **.for** (补全foreach语句)
  - **.format** (使用String.format()包裹一个字符串)
  - **.cast** (使用类型转化包裹一个表达式)

45. 重构（Refactor This）

- **描述：**该操作可以显示所有对当前选中项可行的重构方法。这个列表可以用数字序号快速选择。
- **快捷键：**Ctrl + T(OS X)、Ctrl + Alt + Shift + T(Windows/Linux)

46. 重命名（Rename）

- **描述：**你可以通过该操作重命名变量、字段、方法、类、包。当然了，该操作会确保重命名对下文有意义，不会无脑替换掉所有文件中的名字；
- **快捷键：**Shift + F6
- **更多：**如果你忘记了这个快捷键，你可以使用快速修复（Quick Fix）的快捷键，它通常包含重名选项。

47. 分号/点 补全（Semicolon Dot Completion）

- **描述：**代码补全这个功能太棒啦！我们大概都对以下这种情况很熟悉：开始输入点什么东西，着从IDE得到一些建议的选项，然后通过Enter或者Tab来选择我们想要的补全代码。其实还有另外一方法来选择补全的代码：我们可以输入一个点(.)或者一个分号(;)。这样就会完成补全，添加所选字符这在结束一条语句补全或者快速链式调用方法的时候特别有用。
- **注意点：**如果你要代码补全的方法需要参数，这些参数会被略过。
- **快捷键：**Autocomplete + “.” 或者 “;”

48. 包裹代码（Surround With）

- **描述：** 该操作可以用特定代码结构包裹住选中的代码块，通常是if语句，循环，try/catch语句或者r nnable语句。 如果你没有选中任何东西，该操作会包裹当前一整行。
- **快捷键：**Cmd + Alt + T(OS X)、Ctrl + Alt + T(Windows/Linux)

49. 移除包裹代码（Unwrap Remove）

● **描述：**该操作会移除周围的代码，它可能是一条if语句，一个while循环，一个try/catch语句甚是一个runnable语句。该操作恰恰和包裹代码（Surround With）相反。

● **快捷键：**Cmd + Shift + Delete(OS X)、Ctrl + Shift + Delete(Windows/Linux)

调试技巧

50. 分析传入数据流（Analyze data flow to here）

● **描述：**这个操作将会根据当前选中的变量、参数或者字段，分析出其传递到此处的路径。 当你入某段陌生的代码，试图明白某个参数是怎么传递到此处的时候，这是一个非常有用的操作。

● **调用：**Menu → Analyze → Analyze Data Flow to Here

● **快捷键：**无，可以在设置中指定。

● **相反的操作：**分析传出数据流（Analyze data flow from here），这个将会分析当前选中的变往下传递的路径，直到结束。

51. 堆栈追踪分析（Analyze Stacktrace）

● **描述：** 这个操作读取一份堆栈追踪信息，并且使它像logcat中那样可以点击。当你从bug报告中或终端复制了一份堆栈追踪，使用该操作可以很方便地调试。

● **调用：**Menu → Analyze → Analyze Stacktrace

● **快捷键：**无，可以在设置中指定。

● **更多：**通过使用“ProGuard Unscramble Plugin”插件，也可以分析混淆过的堆栈追踪。

52. 关联调试程序（Attach Debugger）

● **描述：**随时启动调试程序，即使你没有以调试模式启动你的应用。这是一个很方便的操作，因你不必为了调试程序而以调试模式重新部署你的应用。当别人正在测试应用，突然遇到一个bug而将备交给你时，你也可以很快地进入调试模式。

● **调用：**点击工具栏图标或者Menu → Build → Attach to Android Process

● **快捷键：**无，可以在设置中指定，或者点击工具栏对应的图标。

53. 条件断点（Conditional Breakpoints）

● **描述：**简单说，就是当设定的条件满足时，才会触发断点。你可以基于当前范围输入一个java尔表达式，并且条件输入框内是支持代码补全的。

● **调用：**右键需要填写表达式的断点，然后输入布尔表达式。

54. 禁用断点（Disable Breakpoints）

● 这个操作将使得断点。当你有一个设置过复杂条件的断点或者是日志断点，当前不需要，但是下次不用重新创建，该操作是很方便的。

● **调用：**按住Alt，然后单击断点即可。

55. 计算表达式（Evaluate Expression）

● **描述：**这个操作可以用来查看变量的内容并且计算几乎任何有效的java表达式。需要注意的是如果你修改了变量的状态，这个状态在你恢复代码执行后依然会保留。

● **快捷键：**处在断点状态时，光标放在变量处，按Alt + F8，即可显示计算表达式对话框。

56. 审查变量（Inspect Variable）

- **描述：**该操作可以在不打开计算表达式对话框就能审查表达式的值。
- **快捷键：**调试状态下，按住Alt键，然后单击表达式即可。

57. 日志断点（Logging Breakpoints）

- **描述：**这是一种打印日志而不是暂停的断点，当你想打印一些日志信息但是不想添加 log代码重新部署项目，这是一个非常有用的操作。
- **调用：**在断点上右键，取消 Suspend的勾选，然后勾选上Log evaluated Expression，并在入框中输入你要打印的日志信息。

58. 标记对象（Mark Object）

- **描述：**当你在调试的时候，这个操作可以让你给某个特殊的对象添加一个标签，方便你后面很地辨认。在调试时，当你从一堆相似的对象中查看某个对象是否和之前是一样的，这就是一个非常有的操作。
- **调用：**右键你需要标记的对象，选中 Mark Object，输入标签；
- **快捷键：**选中对象时，按F3(OS X)、F11(Windows/Linux)；

59. 显示当前运行点（Show Execution Point）

- **描述：**该操作会立刻把你的光标移回到当前debug处。

通常的情况是： 1. 你在某处触发了断点 2.
然后在文件中随意浏览 3. 直接调用这个快捷键，快速返回之前逐步调试的地
。

- **快捷键：** （Debug时) Alt + F10；

60. 终止进程（Stop Process）

- **描述：**该操作会终止当前正在运行的任务。如果任务数量大于一，则显示一个列表供你选择。终止调试或者中止编译的时候特别有用！
- **快捷键：**Cmd + F2(OS X)、Ctrl + F2（Windows、Linux）；

61. 临时断点（Temporary Breakpoints）

- **描述：**通过该操作可以添加一个断点，这个断点会在第一次被命中的时候自动移除。
- **快捷键：**Alt + 鼠标左键 点击代码左侧（鼠标）、Cmd + Alt + Shift + F8(OS X)、Ctrl + Alt + Shift + F8(Windows/Linux)

62. 调用层级树弹窗（The Call Hierarchy Popup）

- **描述：**该操作会给你展示 在一个方法的声明和调用之间所有可能的路径。
- **快捷键：**Ctrl + Alt + H

63.找到当前类/方法被引用的位置

- **描述：**该操作会找到类/方法被引用的所有位置。

- **快捷键：**Alt + F7、右键 -> Find Usages

63.查找项目中方法或者变量

- **描述：** 找到某方法/变量在项目中的位置。
- **快捷键：Ctrl + Alt + Shift + N**
