我的VPS下tomcat路径：

/usr/tomcat/ap/webapps/ROOT/

/usr/tomcat/ap/bin/shutdown.sh

/usr/tomcat/ap/bin/startup.sh

Android中的activity如果开启了子线程，即使当前此activity被finish，子线程如果没有执行完毕，也依旧会在后台继续执行。有时候会产生bug，因此最好的解决方法是在run方法中的while循环添加变量控制，while(run)，退出activity时将run=false即可。

activity默认在屏幕横竖切换时会reset，即从头开始执行，对正在运行的进程来说是非常不合理的，修改这种情况的方法是在manifest文件，在该activity的属性中添加android:configChanges="orientation|keyboardHidden|screenSize" 即可。

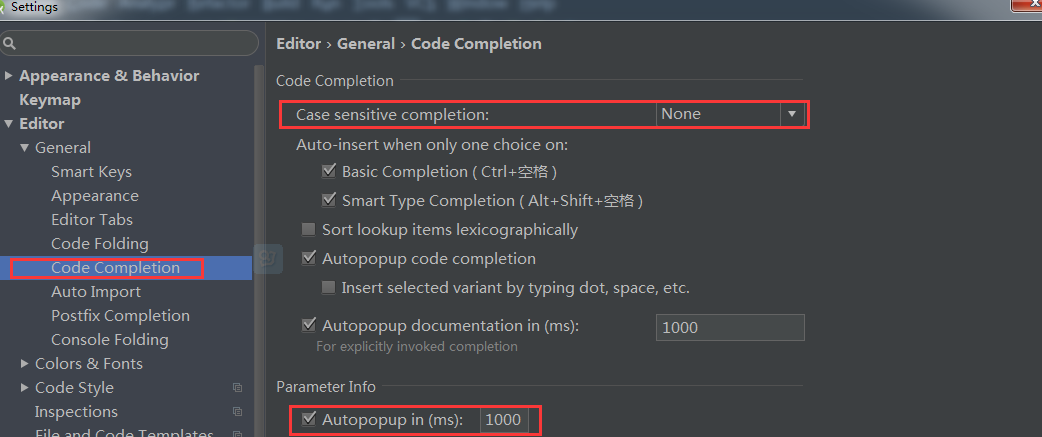
在.xml文件中，引用属性时，@开头表示引用系统公共资源，？开头表示引用当前主题的资源。

SDK镜像地址：

1. 启动 Android SDK Manager ，打开主界面，依次选择**「Tools」**、**「Options...」**，弹出**『Android SDK Manager - Settings』**窗口；
2. 在**『Android SDK Manager - Settings』**窗口中，在**「HTTP Proxy Server」**和**「HTTP Proxy Port」**输入框内填入mirrors.neusoft.edu.cn和80，并且选中**「Force https://... sources to be fetched using http://...」**复选框。设置完成后单击**「Close」**按钮关闭**『Android SDK Manager - Settings』**窗口返回到主界面；
3. 依次选择**「Packages」**、**「Reload」**。

AS中添加函数或者类的注释的快捷键设置是fix doc comment，在keymap中可以自定义新的快捷键。

代码注释时不区分大小写：



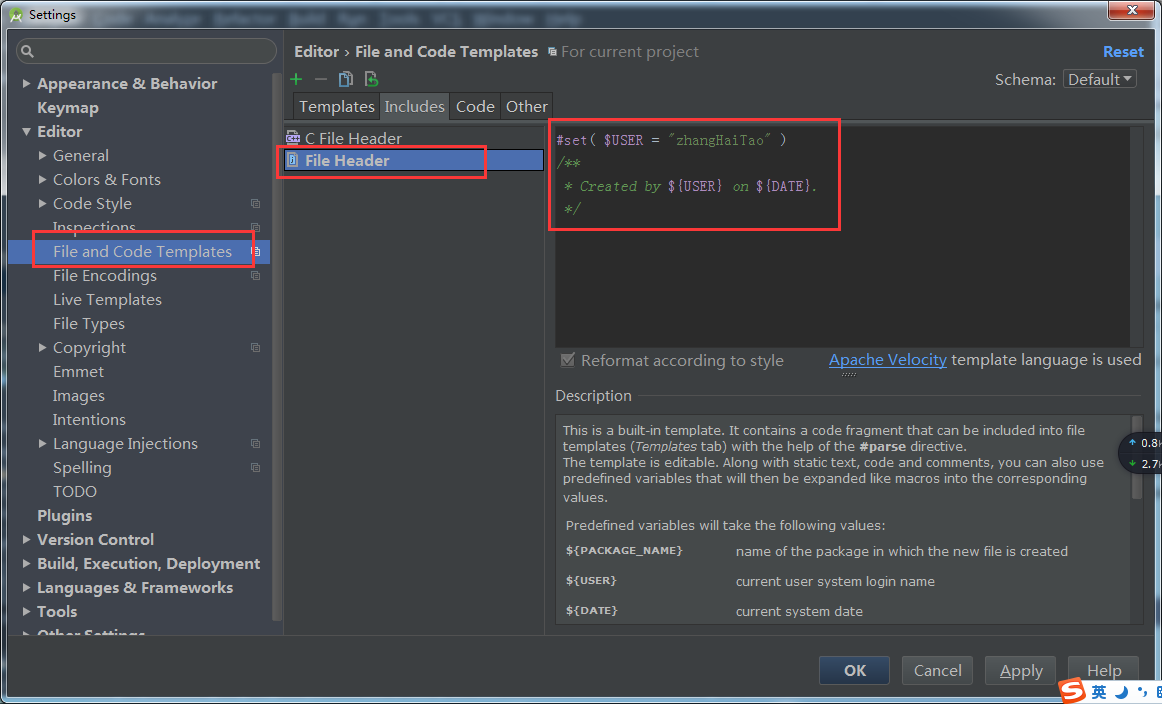
Android Studio下新建类的注释设置：

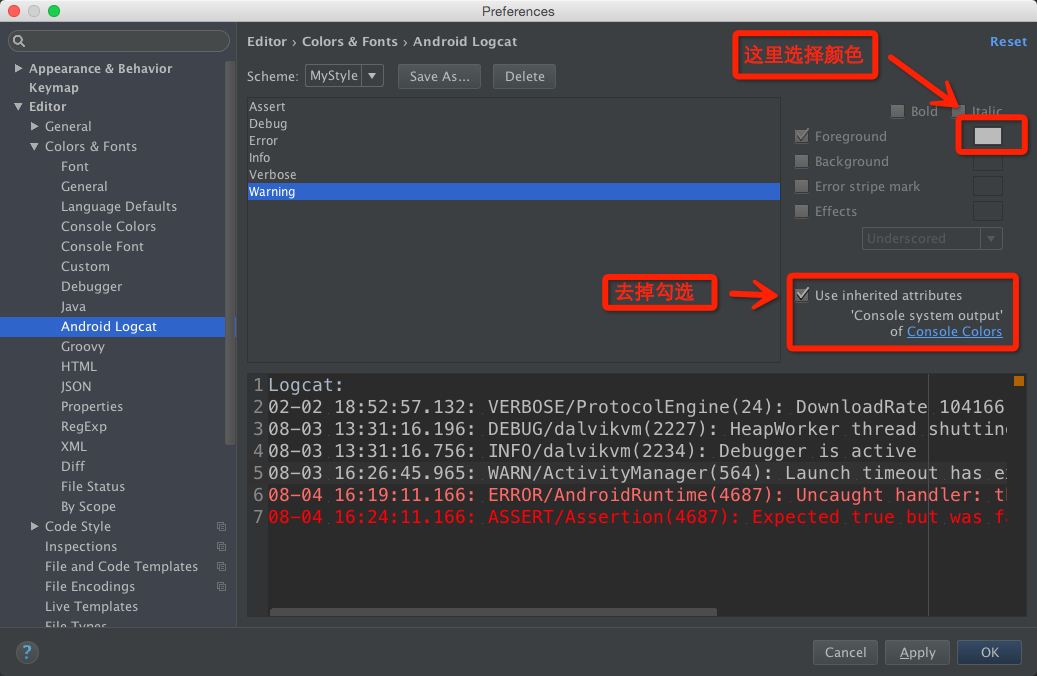
#set( $USER = "zhangHaiTao" )

/\*\*

\* Created by ${USER} on ${DATE}.

\*/



修改Logcat颜色

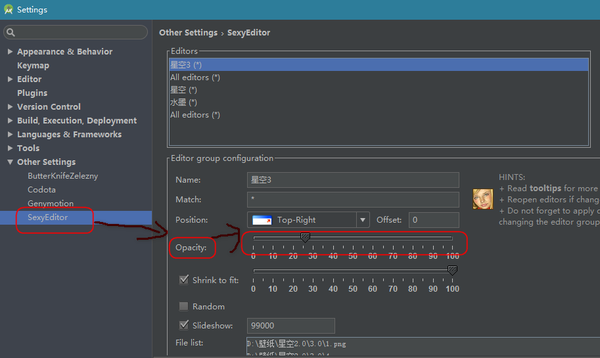
| **Log级别** | **色值** |
| --- | --- |
| VERBOSE | BBBBBB |
| DEBUG | 0070BB |
| INFO | 48BB31 |
| WARN | BBBB23 |
| ERROR | FF0006 |
| ASSERT | 8F0005 |

背景图片设置：

https://www.zhihu.com/question/38958773

插件地址：https://plugins.jetbrains.com/plugin/1833?pr=idea

position可以设置为center



# 导入jar包、module等

<http://blog.csdn.net/eclipsexys/article/details/41142907>

<http://rocko.xyz/2014/12/13/Android-Studio-jar%E3%80%81so%E3%80%81library%E9%A1%B9%E7%9B%AE%E4%BE%9D%E8%B5%96/>

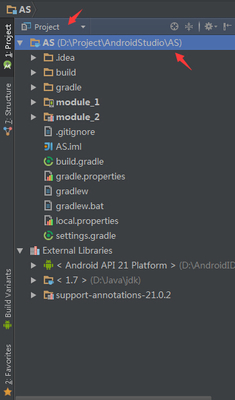
1. 导入 jar 包的方式非常简单，就是在项目中的 libs 中放入你需要导入的 jar 包，然后右键你的 jar 文件，选择“add as a library”即可在你的项目中使用这个 jar 包了。
2. 添加远程开源库,可以无需将开源库下载下来，进入你的项目（一般是 app 目录），编辑 build.gradle 文件。添加依赖如：

compile 'com.google.code.gson:gson:2.6.2'

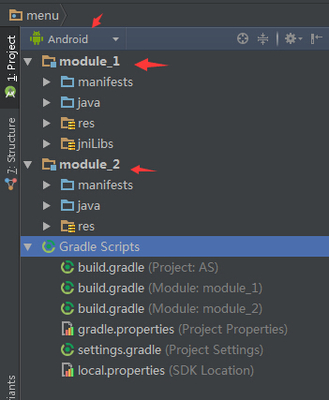
**Eclipse跟AS的不同**

从Eclipse到AS不要带着在Eclipse中的主观色彩去在AS中使用，从项目的构成到构建是不同的，下面列举在Eclipse和AS中的一些概念的区别：

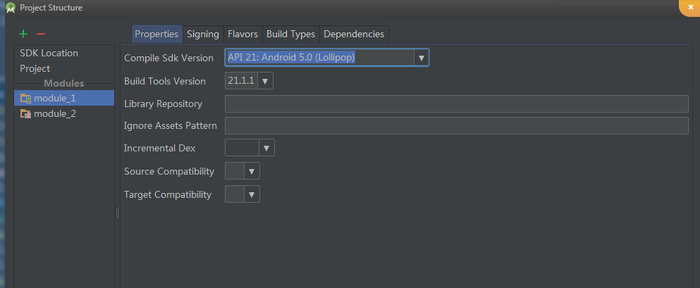
**WorkSpace和Project**

Eclipse的WorkSpace和AS的Project说的可以说是一个东西，也就是说你可以把在AS中的Project理解为WorkSpace。  
[](http://7sblw9.com1.z0.glb.clouddn.com/Android-Studio-jar%E3%80%81so%E3%80%81library%E9%A1%B9%E7%9B%AE%E4%BE%9D%E8%B5%96_1.jpg?imageView2/2/w/400/h/400/q/100)所以你在AS中new一个Project相当于在Eclipse中重开了一个WorkSpace，注意第一个箭头，显示模式为Project，建议刚用AS时用这种，方便了解里面的文件结构。

**Project和Module**

跟上面一样，Eclipse中的一个个project也就是相当于AS中的一个个module。上图的module\_1和module\_2就是我们习惯的eclipse中的一堆project了，把显示模式换为Android之后就更为直观了：  
[](http://7sblw9.com1.z0.glb.clouddn.com/Android-Studio-jar%E3%80%81so%E3%80%81library%E9%A1%B9%E7%9B%AE%E4%BE%9D%E8%B5%96_2.jpg?imageView2/2/w/400/h/400/q/100)  
最下面的就是AS中整个Project中所有Gradle的配置了，当然包括所有module的配置了，括号的名字就表示build.gradle对应的配置对象。

**Properties和Module Setting**

Eclipse中的Properties也是跟AS的Module Setting对应的  
[](http://7sblw9.com1.z0.glb.clouddn.com/Android-Studio-jar%E3%80%81so%E3%80%81library%E9%A1%B9%E7%9B%AE%E4%BE%9D%E8%B5%96_3.jpg?imageView2/2/w/700/h/700/q/100)  
可以看到这里可以像在Eclipse的Properties中一样在这里配置一些东西，比如在Module Setting里给Module添加依赖(dependencies)信息也是可以的，并且可以直接搜maven的项目依赖。

**jar**

明白了上面的三点就可以很快上手了。首先就来说最简单的添加jar包。

* 可以跟在Eclipse中一样，把jar包往Module里扔，再在jar右键add as library就可以了，然后最后在你的Module文件夹（像上面的module\_1）右键make module一下就可以在代码里用jar里的东西了。
* 也可以自己手动到module里的build.gradle里添加dependencies，上面的方法做的方法本质上就是这种。

|  |  |
| --- | --- |
| 1 2 3 4 | dependencies {  compile fileTree(dir: 'libs', include: ['\*.jar'])  compile fileTree(dir: 'D:\\repositories\\libs\\java', include: ['\*.jar']) } |

dir可以是电脑上的目录文件。

**library项目**

有了前面跟Eclipse的比较后，类似地像在Eclipse中添加项目依赖一样，被依赖的项目得是作为library。在Eclipse中我们是进入到Properties把这个项目设置为library（as a library），所以在AS中也是类似的，我们需要把一个module作为library(这个module可以自己新建module也可以导入module，此外我们是可以把一个AS的Project导进成module的或者直接导Project里的单个module也可以)，完成后到这个module（我这里是把module\_2作为library）把apply plugin: 'com.android.application'改为apply plugin: 'com.android.library'再然后去掉（删除）module\_2的build.gradle里的applicationId "com.example.mrzheng.as"(一个library不需要这个，不然make project或make module时会报错)。  
build.gradle(module\_2)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | apply plugin: 'com.android.library' android {  compileSdkVersion 21  buildToolsVersion "21.1.1"  defaultConfig {  minSdkVersion 10  targetSdkVersion 21  versionCode 1  versionName "1.0"  }  buildTypes {  release {  minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  }  } } dependencies {  compile fileTree(dir: 'libs', include: ['\*.jar'])  compile 'com.android.support:appcompat-v7:21.0.2' } |

到这里先确认下你的project(AS)的settings.gradle里有没把module都include进去,没有的话加上：

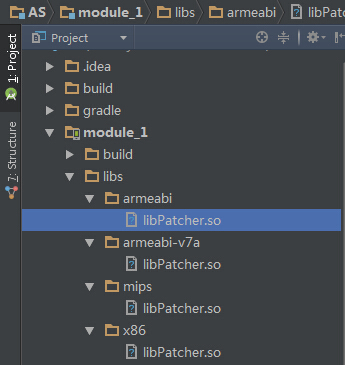
|  |  |
| --- | --- |
| 1 | include ':module\_1', ':module\_2' |

最后就可以在module\_1里就添加library依赖(module\_2)了。进入module\_1的build.gradle，找到dependencies加上compile project(':module\_2')  
build.gradle(module\_1)

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 | apply plugin: 'com.android.application' android {  compileSdkVersion 21  buildToolsVersion "21.1.1"  defaultConfig {  applicationId "com.example.mrzheng.as"  minSdkVersion 10  targetSdkVersion 21  versionCode 1  versionName "1.0"  }   buildTypes {  release {  minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  }  }  sourceSets {  main {  jniLibs.srcDirs = ['libs']  }  }  } dependencies {  compile fileTree(dir: 'libs', include: ['\*.jar'])  compile fileTree(dir: 'D:\\repositories\\libs\\java', include: ['\*.jar'])  compile 'com.android.support:appcompat-v7:21.0.2'  compile project(':module\_2') } |

现在make module一下就可以使用依赖的项目了(module\_2)。

**so**

之前的版本不知道怎么样，现在正式版的AS添加so打包进apk里的lib里是很简单的，我们只需要把so文件放到libs文件夹里的对应cpu文件夹里，最后在module的build.gradle里加上jni的sourceSets配置：jniLibs.srcDirs = ['libs']，完整代码看上面的build.gradle(module\_1)代码片。  
[](http://7sblw9.com1.z0.glb.clouddn.com/Android-Studio-jar%E3%80%81so%E3%80%81library%E9%A1%B9%E7%9B%AE%E4%BE%9D%E8%B5%96_4.jpg?imageView2/2/w/400/h/400/q/100)

**END**

# AS快捷键：

根据构造函数，自动生成变量名声明，快捷键为：Ctrl+Alt+V

| Alt + Enter | 在xml中是提取资源，如从布局文件中提取字符串到strings.xml文件中  修复代码错误 |
| --- | --- |

给方法和类添加注释：



代码提示：



Alt+Insert：生成get,set，构造方法，使用ButterKnife抽取布局文件中的控件。

Ctrl(Command)+F12 显示当前文件的结构

Ctrl(Command)+H 显示类继承结构图

Ctrl(Command)+ ] / [ 跳转到代码块结束/开始处

Ctrl(Command)+Z 倒退

Ctrl(Command)+Shift+Z 向前

Ctrl(Command)+Alt(Option)+T 把选中的代码放在 try{} 、if{} 、 else{} 里

Shift+Enter 在当前行的下面插入新行，并移动光标到新行

Ctrl(Command)+Enter 在当前行的上面插入新行，并移动光标到新行（此功能光标在行首时有效）

Ctrl(Command)+Alt(Option)+L 格式化代码

Ctrl(Command)+Alt(Option)+I 将选中的代码进行自动缩进编排

Ctrl(Command)+Alt(Option)+O 优化导入的类和包

Ctrl+O：覆盖方法

Ctrl+Q：查看方法的注释文档。

Ctrl+P:提示方法的参数信息

Ctrl+X：删除一行

Ctrl+E:最近打开的文件

Ctrl+H:显示类结构图

F2或Shift+F2:在高亮错误或警告快速定位

Ctrl+Shift+C:对比最近修改的代码

Alt(Option)+Left/Right 切换代码视图

Ctrl+Alt+Left/Right:返回至上次浏览的位置

Ctrl(Command)+Shift+Back 跳转到上次编辑的地方

Ctrl(Command)+Shift+U 大小写切换

Ctrl(Command)+Backspace 按单词删除

Ctrl(Command)+B 统计变量使用情况 快速打开光标处的类或方法

**附录（IDEA(Android Studio) 快捷键）**

说明：斜体文字表示，测试时没有效果或者没有测试时没有达到预先条件的情况下没有效果。

## IDE

| **按键** | **说明** |
| --- | --- |
| F1 | 帮助 |
| Alt+F1 | 查找文件所在目录位置 |
| Alt+1 | 快速打开或隐藏工程面板 |
| Ctrl+Alt+S | 打开设置对话框 |
| Alt+Home | 跳转到导航栏 |
| Esc | 光标返回编辑框 |
| Shift+Esc | 光标返回编辑框,关闭无用的窗口 |
| Shift+Click | 关闭标签页 |
| F12 | 把焦点从编辑器移到最近使用的工具窗口 |
| Ctrl+Alt+Y | 同步 |
| Ctrl+Alt+S | 打开设置对话框 |
| Alt+Shift+Inert | 开启/关闭列选择模式 |
| Ctrl+Alt+Shift+S | 打开当前项目/模块属性 |
| Alt+Shift+C | 查看文件的变更历史 |
| Ctrl+Shift+F10 | 运行 |
| Ctrl+Shift+F9 | debug运行 |
| Ctrl+Alt+F12 | 资源管理器打开文件夹 |

## 编辑

| **按键** | **说明** |
| --- | --- |
| Alt+J | 多行编辑 |
| Alt+MouseDrag | 鼠标选中区域 |
| Ctrl+C | 复制当前行或选中的内容 |
| Ctrl+D | 粘贴当前行或选中的内容到下一行 |
| Ctrl+X | 剪切当前行或选中的内容 |
| Ctrl+Y | 删除行 |
| Ctrl+Z | 倒退 |
| Ctrl+Shift+Z | 向前 |
| Alt+Enter | 自动修正 |
| Ctrl+Alt+L | 格式化代码 |
| Ctrl+Alt+I | 将选中的代码进行自动缩进编排 |
| Ctrl+Alt+O | 优化导入的类和包 |
| Ctrl+Alt+D | 对选中内容添加包围代码 |
| Ctrl+Alt+J | 用动态模板环绕 |
| Alt+Insert | 得到一些Intention Action，可以生成构造器、Getter、Setter、将 == 改为 equals() 等 |
| Ctrl+Shift+V | 选最近使用的剪贴板内容并插入 |
| Ctrl+Alt+Shift+V | 简单粘贴 |
| Ctrl+Shift+Insert | 选最近使用的剪贴板内容并插入（同Ctrl+Shift+V） |
| Ctrl+Enter | 在当前行的上面插入新行，并移动光标到新行（此功能光标在行首时有效） |
| Shift+Enter | 在当前行的下面插入新行，并移动光标到新行 |
| Ctrl+J | 自动代码 |
| Ctrl+Alt+T | 把选中的代码放在 try{} 、if{} 、 else{} 里 |
| Shift+Alt+Insert | 竖编辑模式 |
| Ctrl+ / | 注释 // |
| Ctrl+Shift+ / | 注释 /\*...\*/ |
| Ctrl+Shift+J | 合并成一行 |
| F2/Shift+F2 | 跳转到下/上一个错误语句处 |
| Ctrl+Shift+Back | 跳转到上次编辑的地方 |
| Ctrl+Alt+Space | 类名自动完成 |
| Shift+Alt+Up/Down | 内容向上/下移动 |
| Ctrl+Shift+Up/Down | 语句向上/下移动 |
| Ctrl+Shift+U | 大小写切换 |
| Tab | 代码标签输入完成后，按 Tab，生成代码 |
| Ctrl+Backspace | 按单词删除 |
| Ctrl+Shift+Enter | 语句完成 |

## 文件

| **按键** | **说明** |
| --- | --- |
| Ctrl+F12 | 显示当前文件的结构 |
| Ctrl+H | 显示类继承结构图 |
| Ctrl+Q | 显示注释文档 |
| Ctrl+P | 方法参数提示 |
| Ctrl+U | 打开当前类的父类或者实现的接口 |
| Alt+Left/Right | 切换代码视图 |
| Ctrl+Alt+Left/Right | 返回上次编辑的位置 |
| Alt+Up/Down | 在方法间快速移动定位 |
| Ctrl+B | 快速打开光标处的类或方法 |
| Ctrl+W | 选中代码，连续按会有其他效果 |
| Ctrl+Shift+W | 取消选择光标所在词 |
| Ctrl+ - / + | 折叠/展开代码 |
| Ctrl+Shift+ - / + | 折叠/展开全部代码 |
| Ctrl+Shift+. | 折叠/展开当前花括号中的代码 |
| Ctrl+ ] / [ | 跳转到代码块结束/开始处 |
| F2 或 Shift+F2 | 高亮错误或警告快速定位 |
| Ctrl+Shift+C | 复制路径 |
| Ctrl+Alt+Shift+C | 复制引用，必须选择类名 |
| Alt+Up/Down | 在方法间快速移动定位 |
| Shift+F1 | 要打开编辑器光标字符处使用的类或者方法 Java 文档的浏览器 |
| Ctrl+G | 定位行 |

## 查找

| **按键** | **说明** |
| --- | --- |
| Ctrl+F | 在当前窗口查找文本 |
| Ctrl+Shift+F | 在指定环境下查找文本 |
| F3 | 向下查找关键字出现位置 |
| Shift+F3 | 向上一个关键字出现位置 |
| Ctrl+R | 在当前窗口替换文本 |
| Ctrl+Shift+R | 在指定窗口替换文本 |
| Ctrl+N | 查找类 |
| Ctrl+Shift+N | 查找文件 |
| Ctrl+Shift+Alt+N | 查找项目中的方法或变量 |
| Ctrl+B | 查找变量的来源 |
| Ctrl+Alt+B | 快速打开光标处的类或方法 |
| Ctrl+Shift+B | 跳转到类或方法实现处 |
| Ctrl+E | 最近打开的文件 |
| Alt+F3 | 快速查找，效果和Ctrl+F相同 |
| F4 | 跳转至定义变量的位置 |
| Alt+F7 | 查询当前元素在工程中的引用 |
| Ctrl+F7 | 查询当前元素在当前文件中的引用，然后按 F3 可以选择 |
| Ctrl+Alt+F7 | 选中查询当前元素(方法或元素)在工程中的引用 |
| Ctrl+Alt+H | 打开方法调用结构层级的窗口 |
| Ctrl+Shift+F7 | 高亮显示匹配的字符，按 Esc 高亮消失 |
| Ctrl+Shift+Alt+N | 查找类中的方法或变量 |
| Ctrl+F12 | 查找类中的方法 |
| Ctrl+Shift+O | 弹出显示查找内容 |
| Ctrl+Alt+Up/Down | 快速跳转搜索结果 |
| Ctrl+Shift+S | 高级搜索、搜索结构 |

## 重构

| **按键** | **说明** |
| --- | --- |
| F5 | 复制 |
| F6 | 移动 |
| Alt+Delete | 安全删除 |
| Ctrl+U | 转到父类 |
| Ctrl+O | 重写父类的方法 |
| Ctrl+I | 实现方法 |
| Ctrl+Alt+N | 内联 |
| Ctrl+Alt+Shift+T | 弹出重构菜单 |
| Shift+F6 | 重构-重命名 |
| Ctrl+Alt+M | 提取代码组成方法 |
| Ctrl+Alt+C | 将变量更改为常量 |
| Ctrl+Alt+V | 定义变量引用当前对象或者方法的返回值 |
| Ctrl+Alt+F | 将局部变量更改为类的成员变量 |
| Ctrl+Alt+P | 将变量更改为方法的参数 |

## 调试

| **按键** | **说明** |
| --- | --- |
| F8 | 跳到下一步 |
| Shift+F8 | 跳出函数、跳到下一个断点 |
| Alt+Shift+F8 | 强制跳出函数 |
| F7 | 进入代码 |
| Shift+F7 | 智能进入代码 |
| Alt+Shift+F7 | 强制进入代码 |
| Alt+F9 | 运行至光标处 |
| Ctrl+Alt+F9 | 强制运行至光标处 |
| Ctrl+F2 | 停止运行 |
| Alt+F8 | 计算变量值 |

## VCS

| **按键** | **说明** |
| --- | --- |
| Alt+ ~ | VCS 操作菜单 |
| Ctrl+K | 提交更改 |
| Ctrl+T | 更新项目 |
| Ctrl+Alt+Shift+D | 显示变化 |

## 其它

| **按键** | **说明** |
| --- | --- |
| (Ctrl+)F11 | 标记书签 |
| Shift+F11 | 显示书签 |