

# 目录

|                |           |
|----------------|-----------|
| 前言             | 1.1       |
| 编辑器和IDE简介      | 1.2       |
| 常见编辑器和IDE      | 1.3       |
| 通用功能和逻辑        | 1.4       |
| 编辑器通用功能和逻辑     | 1.4.1     |
| IDE通用功能和逻辑     | 1.4.2     |
| 编辑器和IDE详解      | 1.5       |
| 编辑器详解          | 1.5.1     |
| VSCode         | 1.5.1.1   |
| Sublime Text   | 1.5.1.2   |
| Atom           | 1.5.1.3   |
| vi/vim         | 1.5.1.4   |
| Emacs          | 1.5.1.5   |
| Notepad++      | 1.5.1.6   |
| Source Insight | 1.5.1.7   |
| IDE详解          | 1.5.2     |
| JetBrains公司    | 1.5.2.1   |
| PyCharm        | 1.5.2.1.1 |
| IntelliJ IDEA  | 1.5.2.1.2 |
| PhpStorm       | 1.5.2.1.3 |
| WebStorm       | 1.5.2.1.4 |
| Visual Studio  | 1.5.2.2   |
| Eclipse        | 1.5.2.3   |
| Android Studio | 1.5.2.4   |
| Aptana Studio  | 1.5.2.5   |
| Xcode          | 1.5.2.6   |
| 微信开发者工具        | 1.5.2.7   |
| HBuilder       | 1.5.2.8   |
| 编辑器和IDE总结      | 1.6       |
| 附录             | 1.7       |
| 参考资料           | 1.7.1     |

# 编辑器和IDE总结

- 最新版本： v1.0
- 更新时间： 20190712

## 简介

总结编辑器和IDE的不同角度的分类，常见编辑器和IDE简介，编辑器和IDE通用功能整理，各种常见编程语言的各种IDE，并针对每款常用编辑器和IDE深入详细介绍

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitbook源码

- [crifan/editor\\_ide\\_summary: 编辑器和IDE总结](#)

### 如何使用此Gitbook源码去生成发布为电子书

详见：[crifan/gitbook\\_template: demo how to use crifan gitbook template and demo](#)

### 在线浏览

- [编辑器和IDE总结 book.crifan.com](#)
- [编辑器和IDE总结 crifan.github.io](#)

### 离线下载阅读

- [编辑器和IDE总结 PDF](#)
- [编辑器和IDE总结 ePUB](#)
- [编辑器和IDE总结 MOBI](#)

## 版权说明

此电子书教程的全部内容，如无特别说明，均为本人原创和整理。其中部分内容参考自网络，均已备注了出处。如有发现侵犯您版权，请通过邮箱联系我 `admin 艾特 crifan.com`，我会尽快删除。谢谢合作。

## 鸣谢

感谢我的老婆陈雪的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

## 更多其他电子书

本人 `crifan` 还写了其他 100+ 本电子书教程，感兴趣可移步至：

crifan/crifan\_ebook\_readme: Crifan的电子书的使用说明

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2021-01-16  
15:26:35

# 编辑器和IDE简介

在作为普通电脑用户和技术人员开发期间，往往会用到很多工具软件，去编辑代码文件，开发和调试项目代码。

在此期间，用于编辑文件和调试项目的工具，就是此处要介绍的：

- 文本编辑器 = Text Editor
  - 简称： Editor = 编辑器
  - 主要用途：
    - 普通用户： 编辑文本文件
    - 开发人员： 编写代码
- IDE = Integrated Development Environment = 集成开发环境
  - 主要用途：
    - 开发人员： 编写代码 + 调试代码

对于编辑器和IDE，先总结宏观概况，再解释微观细节：

## 编辑器和IDE的分类

宏观上说，有多种不同角度去分类编辑器和IDE，同时附带简易的举例和说明：

- 根据功能强弱去分
  - 编辑器=Editor
    - 举例
      - Win中最简单的： 记事本
      - Mac中最简单的： 文本编辑
      - Win中相对通用的编辑功能的： Notepad++
      - Win中用于开发方面的查看Linux内核源码的： Source Insight
    - 集成开发环境=IDE
      - 解释：
        - IDE=集成开发环境
        - 集成： 在普通的（编辑文件的）编辑器基础上，额外加上了 其他开发所需功能
          - 谓之： 集成
          - 其他开发所需功能 包含哪些：
            - 编译器= compiler
            - 调试器= debugger
            - 版本控制系统= vcs： 比如 svn , git 等
            - 图形用户界面= GUI
            - 集成其他工具
              - 终端= Terminal
              - 等等
            - 等等
          - 开发环境： 主要用于技术开发领域，调试某些编程语言的代码
        - 举例
          - Win中通用的IDE： Eclipse
          - Mac中开发Python的： PyCharm
      - 额外说明：
        - 现在的Editor和IDE的边界越来越模糊了，尤其像VSCode这种，作为编辑器功能足够强大，加上各种插件后，可以称之为IDE了
    - 根据是否需要网络去分

- 本地的离线的编辑器或IDE
  - 比如目前大多数的编辑器都是安装到本地电脑上即可使用，无需网络
- 在线的云端的编辑器或IDE
  - 比如新出现的一些编辑器或IDE，需要有网络，通过浏览器去使用，属于远程的云端的在线编辑器或IDE
- 额外说明
  - 部分工具，如 `vsCode` 既支持本地离线，又支持云端在线
- 根据支持的平台和系统的多少去分
  - 不少编辑器和IDE，支持多种平台和系统，称为：跨平台
    - 比如
      - `VSCode` 支持Win, Mac, Linux等多种平台
  - 有些编辑器和IDE，只支持特定的平台和系统，不支持跨平台
    - 比如 `Notepad++` 只支持Win平台
- 根据不同编程语言去分
  - 很多编辑器或IDE，是专门针对某些编程语言的
    - 比如
      - 专门只支持一种语言Python的 `Spyder`
  - 有些编辑器或IDE，是相对通用的，支持更多，更广泛的语言
    - 比如
      - `JetBrains` 旗下的 `PyCharm`，虽主要针对Python设计的，但是也支持其他如JS, HTML, CSS等Web领域的开发
- 根据功能是否支持扩展去分
  - 支持扩展：有些通用的编辑器或IDE，通过插件支持更多其他编程语言
    - 比如 `Eclipse`，是个通用的IDE，通过插件 `PyDev` 可以支持调试Python
  - 不支持扩展：其他更多的编辑器或IDE，功能是固定的，往往只针对单个或某个领域的编程语言
    - 比如 `WebStorm`，主要针对于JS语言和相关的Web领域
      - 虽然 `WebStorm` 也支持插件机制，但是主要是安装一些小的功能，其本身的开发和调试的语言，还是侧重在Web领域和JS

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook 最后更新：2019-07-12  
22:59:54

# 常见编辑器和 IDE

下面介绍常见的编辑器和 IDE：

## 编辑器 vs IDE

- 编辑器

- 跨平台

- VSCode = VS Code = Visual Studio Code
    - Sublime = Sublime Text
    - Atom
    - GEdit

- Linux类

- vi
      - 增强版： vim
    - Emacs
    - GEdit

- Mac

- 文本编辑
    - TextMate

- Windows

- Notepad
    - Notepad++
    - Notepad2
    - EditPlus
    - SlickEdit
    - 专门的：
      - Python
      - IDLE
    - Source Insight

- IDE

- 跨平台

- Eclipse
    - JetBrains 公司



Check out our IDEs



Smart IDE for iOS/OS X development

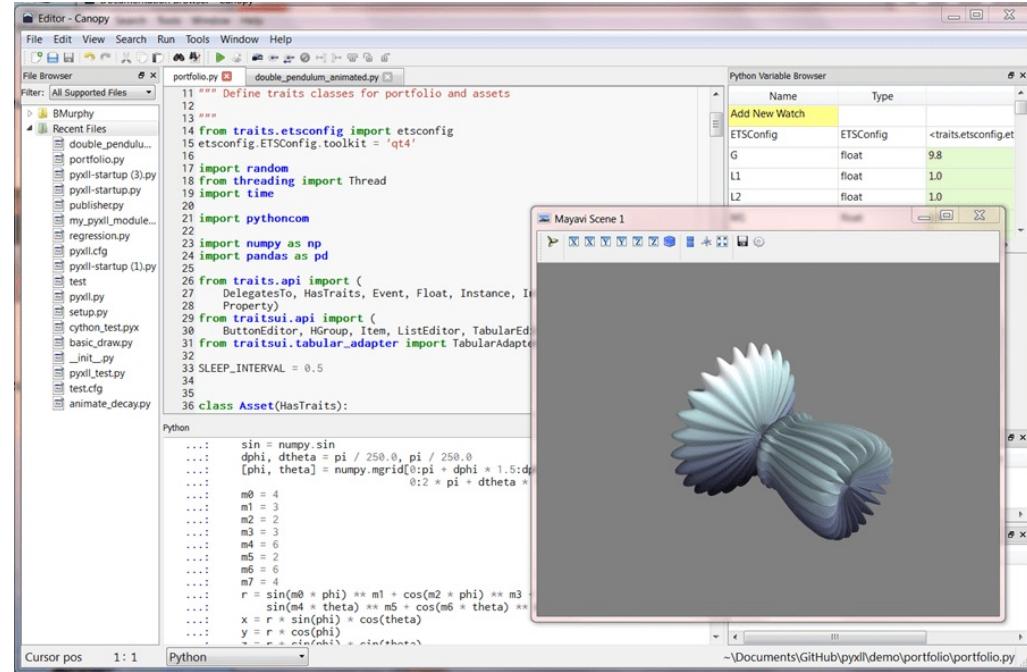
GET FREE 30-DAY TRIAL

- [IntelliJ IDEA](#): for Java
  - 最大优点: 智能提示和自动完成, 重构
  - 最大缺点: 比较耗资源
- 其他基于 IDEA 的 IDE
  - JetBrains 公司自己的
    - [AppCode](#): for iOS/macOS
      - Objective-C
      - Swift
      - C/C++
    - [CLion](#): for C and C++
    - [Goland](#): for Go
    - [PhpStorm](#): for PHP
    - [PyCharm](#): for Python
    - [Rider](#): for .NET
    - [RubyMine](#): for Ruby
    - [WebStorm](#): for Javascript (web 前端)
      - Web 的 Angular/React/Vue.js
      - Mobile 的 Ionic/Cordova/React Native
      - Server 的 Node.js/Meteor
      - Desktop 的 Electron
  - Google 的:
    - [Android Studio](#) : for Android
- [Aptana Studio](#)
- Mac
  - [Xcode](#) : for Mac 的所有平台
  - 微信开发者工具 : for 微信小程序
    - 旧称: 微信web开发者工具
  - [HBuilder](#)
- Windows
  - [VS = Visual Studio](#)
    - C#

## IDE分类

### 根据语言分

- Python
  - [PyCharm](#)
  - [Eclipse+PyDev](#)
  - [Spyder](#)
  - [PyScripter](#)
  - [WingIDE](#)
  - [Enthought Canopy](#)
    - 简介: 面向科学家 和 工程师的 Python IDE, 它预装了为数据分析而用的库
    - 主页: <https://assets.enthought.com/downloads/>
    - 截图:



- Komodo
- Eric
- Java
  - NetBeans
  - JCreator
- C#
  - Visual Studio
- PHP
  - PhpStorm
- Android
  - Android Studio
- 微信小程序
  - 微信开发者工具

## 根据主要适用领域

- Web领域
  - Dreamweaver
  - WebStorm
  - Aptana Studio
  - HBuilder
  - Brackets

## 是否支持云端

- 云端IDE = Web端IDE
  - Web版VSCode
  - Gitee IDE
    - 特点
      - 基于微软的 Monaco Editor
  - CodeSandbox
    - 主要用于: Web前端

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:01:23

# 通用功能和逻辑

此处整理，编辑器和IDE都通用的功能：

## 支持显示特殊字符包括不可见字符

### 字符编码知识

关于不可见字符，控制字符，想要深入了解的，可以去看 [【整理Book】字符编码详解与应用](#) 和 [字符编码简明教程 – 在路上](#) 所提到的：

- ASCII字符集共27个字符
- =128个字符
- = 33个控制字符 + 95个可见字符

和 [字符编码详解](#) 中的[2.1.2.1. ASCII字符集中的功能/控制字符](#)

很多编辑器，支持显示特殊字符，比如：

### Notepad++

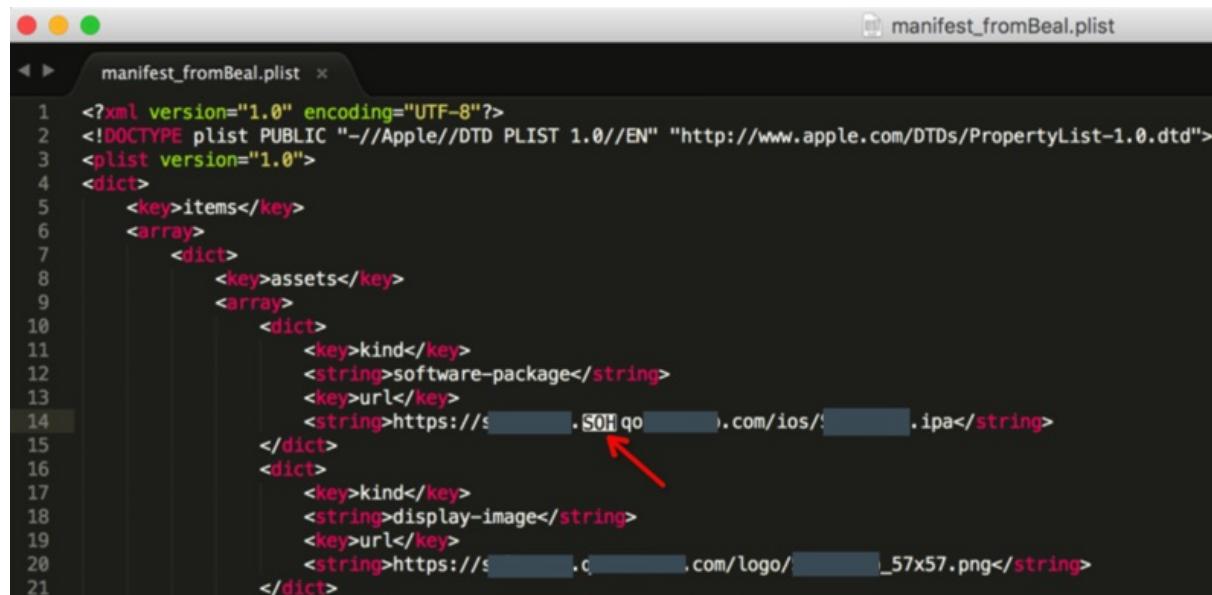
[Notepad++支持显示回车符，换行符，TAB键，行首，行尾等特殊字符](#)

### Sublime

举例：

Sublime在需要的时候，能够显示出特殊的SOH字符：

[\[已解决\] 企业版iOS的ipa通过OTA发布后还是无法下载和安装](#)



A screenshot of the Sublime Text code editor showing a file named "manifest\_fromBeal.plist". The code is an XML-based plist file. A red arrow points to a specific character in the string value of a key. The character is a SOH (Start Of Header) character, which is a control character used in XML to indicate the start of the header section.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>items</key>
    <array>
        <dict>
            <key>assets</key>
            <array>
                <dict>
                    <key>kind</key>
                    <string>software-package</string>
                    <key>url</key>
                    <string>https://s... .SOH qo ... .com/ios/ ... .ipa</string>
                </dict>
                <dict>
                    <key>kind</key>
                    <string>display-image</string>
                    <key>url</key>
                    <string>https://s... .com/logo/ ... _57x57.png</string>
                </dict>
            
```

以协助帮忙找到iOS的OTA版的ipa文件为何无法下载。

### VSCode

详见：

[【已解决】替换掉VSCode中显示出的特殊字符：NAK](#)

## 选中或光标所在位置的内容的高亮显示

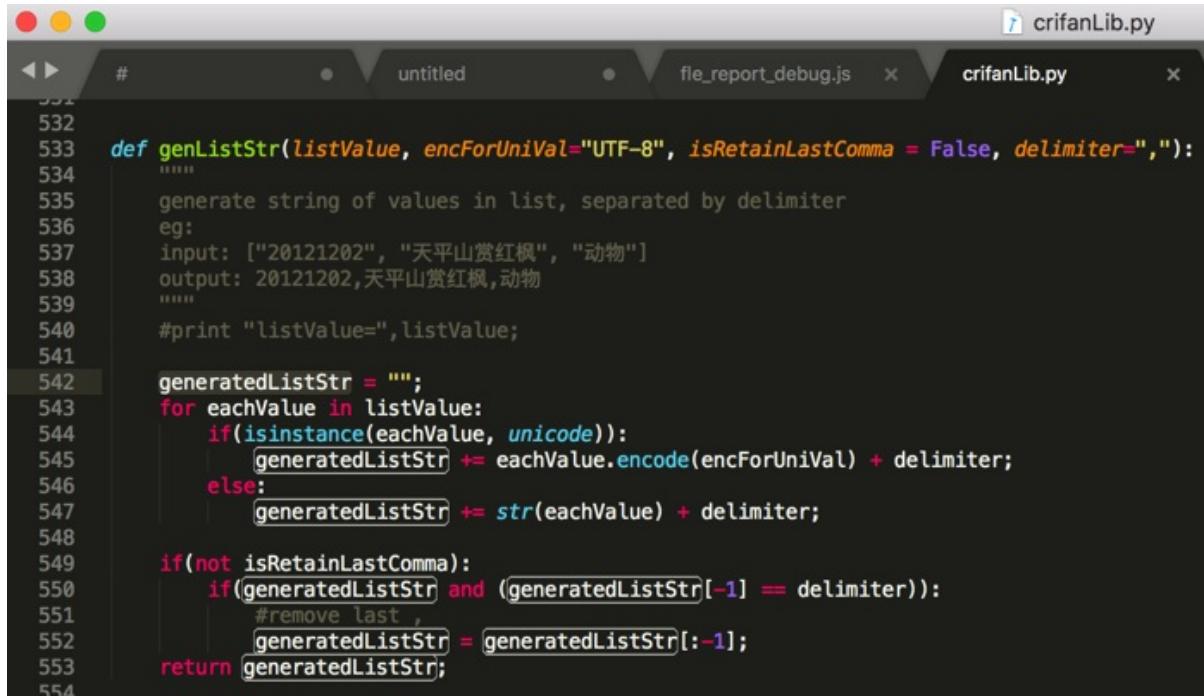
一般的编辑器，IDE都支持常见的：

选中（或者光标所在位置的）一个变量/函数/字符串，则对应的单词会高亮显示

目的是方便及时找到和查看对应的变量/函数等的被使用的地方。

举例：

### Sublime的选中高亮



A screenshot of the Sublime Text editor interface. The title bar shows 'crifanLib.py'. Below the title bar, there are tabs for '#', 'untitled', 'file\_report\_debug.js', and 'crifanLib.py'. The main code area displays Python code. In line 542, the variable 'generatedListStr' is highlighted in red, indicating it is currently selected. The code is as follows:

```

532
533     def genListStr(listValue, encForUniVal="UTF-8", isRetainLastComma = False, delimiter=","):
534         """
535             generate string of values in list, separated by delimiter
536             eg:
537                 input: ["20121202", "天平山赏红枫", "动物"]
538                 output: 20121202,天平山赏红枫,动物
539         """
540         #print "listValue=",listValue;
541
542         generatedListStr = "";
543         for eachValue in listValue:
544             if(isinstance(eachValue, unicode)):
545                 generatedListStr += eachValue.encode(encForUniVal) + delimiter;
546             else:
547                 generatedListStr += str(eachValue) + delimiter;
548
549             if(not isRetainLastComma):
550                 if(generatedListStr and (generatedListStr[-1] == delimiter)):
551                     #remove last ,
552                     generatedListStr = generatedListStr[:-1];
553
554         return generatedListStr;

```

### VSCode中选中高亮

The screenshot shows a code editor in Eclipse with several tabs at the top: 'BlogsToWordpress.py', 'BlogCsdn.py', 'crifanLib.py', and 'cmd\_example.txt'. The main pane displays Python code for a function named 'getUrlResponse'. A tooltip 'dict' appears over the word 'postData', indicating it is being used as a dictionary. The code uses the 'urllib2' module to handle URLs and POST requests.

```

1209     return isdownOK;
1210
1211     #-----
1212 def getUrlResponse(url, postDict={}, headerDict={}, timeout=0, useGzip=False, postDataDelimiter."&") :
1213     """Get response from url, support optional postDict,headerDict,timeout,useGzip
1214
1215     Note:
1216     1. if postDict not null, url request auto become to POST instead of default GET
1217     2. if you want to auto handle cookies, should call initAutoHandleCookies() before use this function
1218     | then following urllib2.Request will auto handle cookies
1219     """
1220
1221     # makesure url is string, not unicode, otherwise urllib2.urlopen will error
1222     url = str(url);
1223
1224     if (postDict) :
1225         if(postDataDelimiter=="&"):
1226             postData = urllib.urlencode(postDict);
1227         else:
1228             postData = "";
1229             for eachKey in postDict.keys() :
1230                 postData += str(eachKey) + "=" + str(postDict[eachKey]) + postDataDelimiter;
1231     postData = postData.strip();
1232     #logging.info("postData=%s", postData);
1233     req = urllib2.Request(url, postData);
1234     #logging.info("req=%s", req);
1235     req.add_header('Content-Type', "application/x-www-form-urlencoded");
1236     else :
1237         req = urllib2.Request(url);

```

## Eclipse的选中高亮

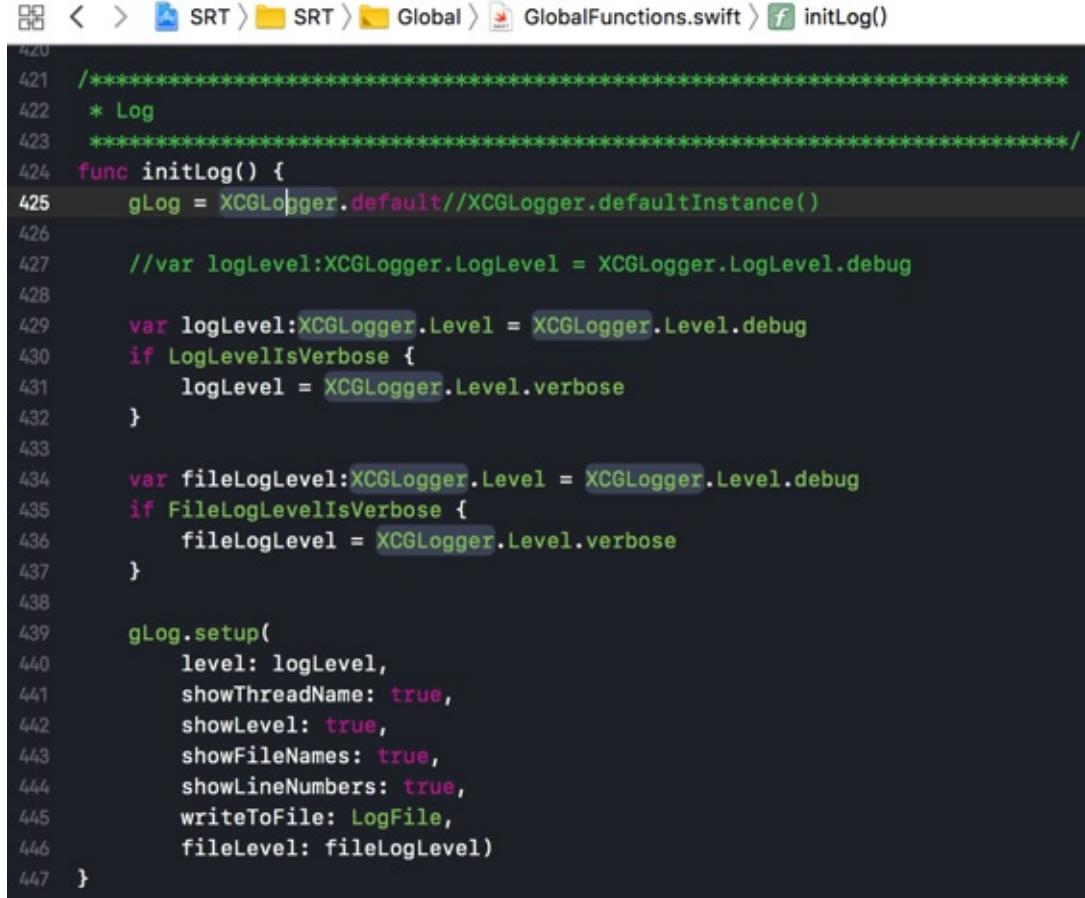
The screenshot shows a code editor in Eclipse with several tabs at the top: 'CreateCustomerRuleCheck.java', 'ScriptTriggerParam.java', 'ScriptBusinessException.java', and 'ScriptBusinessException.java'. The main pane displays Java code for a class 'CreateCustomerRuleCheck' that implements 'ScriptTrigger'. A tooltip 'Logger' appears over the variable 'logger', indicating it is being used as a logger. The code uses the 'ScriptEngineManager' and 'ScriptEngine' classes.

```

12 public class CreateCustomerRuleCheck implements ScriptTrigger {
13     private static final Logger logger = LoggerFactory.getLogger();
14
15     @Override
16     public ScriptTriggerResult execute(ScriptTriggerParam scriptTri
17         if (scriptTriggerParam != null) {
18             // 获取业务对象列表
19             List<DataModel> modelList = scriptTriggerParam.getDataMode
20             if (modelList != null & modelList.size() > 0) {
21                 for (DataModel dataModel : modelList) {
22                     // 获取业务对象备注信息；字段信息可通过业务对象的describe
23                     String comment = (String) dataModel.getAttribute("c
24 //                     if ("forbid".contains(comment)) {
25                     if (comment.contains("forbid")) {
26                         // 记录错误信息，可在销售易后台业务逻辑代码日志中查询/
27                         logger.error("this account is not allowed to s
28                         throw new ScriptBusinessException("新建客户失败:
29                     }
30                 }
31             }
32         }

```

## Xcode的选中高亮



```

420
421  ****
422  * Log
423  ****
424 func initLog() {
425     gLog = XCGLogger.default//XCGLogger.defaultInstance()
426
427     //var logLevel:XCGLogger.LogLevel = XCGLogger.LogLevel.debug
428
429     var logLevel:XCGLogger.Level = XCGLogger.Level.debug
430     if LogLevelIsVerbose {
431         logLevel = XCGLogger.Level.verbose
432     }
433
434     var fileLogLevel:XCGLogger.Level = XCGLogger.Level.debug
435     if FileLogLevelIsVerbose {
436         fileLogLevel = XCGLogger.Level.verbose
437     }
438
439     gLog.setup(
440         level: logLevel,
441         showThreadName: true,
442         showLevel: true,
443         showFileNames: true,
444         showLineNumbers: true,
445         writeToFile:LogFile,
446         fileLevel: fileLogLevel)
447 }

```

## 列编辑模式

有些时候要多列一起编辑，则支持列编辑模式的话，就很好用。

支持列编辑的编辑器或IDE：

- Notepad++
- VSCode

### VSCode支持列编辑

- 批量删除行首的内容
  - Alt+Shift+左键 选择：

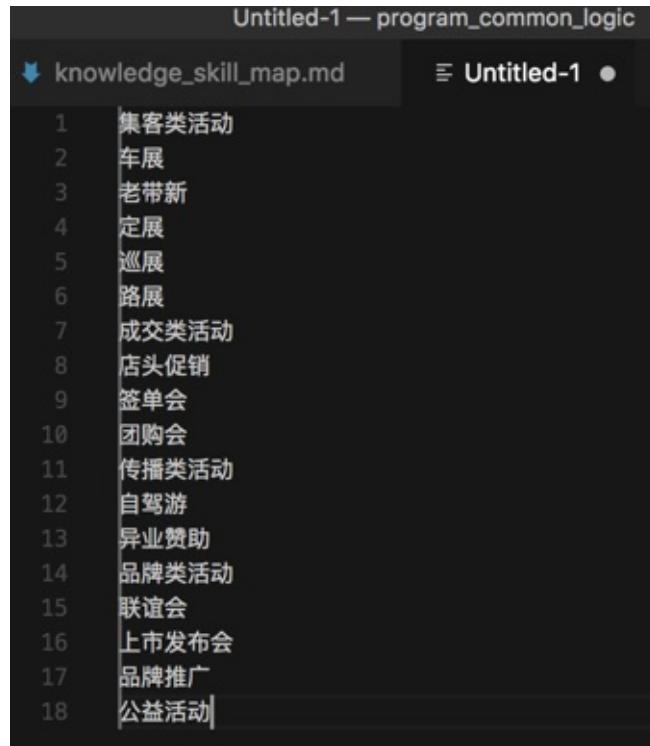


Untitled-1 — program\_common\_logic

knowledge\_skill\_map.md

```
1  □ 集客类活动
2  □ 车展
3  □ 老带新
4  □ 定展
5  □ 巡展
6  □ 路展
7  □ 成交类活动
8  □ 店头促销
9  □ 签单会
10 □ 团购会
11 □ 传播类活动
12 □ 自驾游
13 □ 异业赞助
14 □ 品牌类活动
15 □ 联谊会
16 □ 上市发布会
17 □ 品牌推广
18 □ 公益活动
19
```

- 然后删除掉：

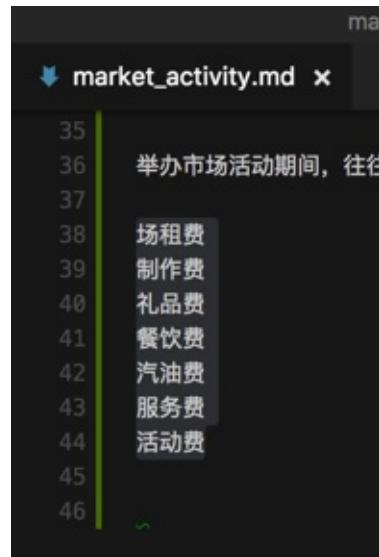


Untitled-1 — program\_common\_logic

knowledge\_skill\_map.md

```
1  集客类活动
2  车展
3  老带新
4  定展
5  巡展
6  路展
7  成交类活动
8  店头促销
9  签单会
10 团购会
11 传播类活动
12 自驾游
13 异业赞助
14 品牌类活动
15 联谊会
16 上市发布会
17 品牌推广
18 公益活动
```

- 即可批量删除每行前面的多余内容了
- 批量给行首添加内容
  - 想要给每行前面添加星号和空格
    - 先选择要编辑的多行的内容



```
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46 举办市场活动期间，往往
```

- 鼠标点击到开始位置，然后 Alt+Shift+鼠标左键 选择到所有的行的行首：



```
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46 举办市场活动期间，往往
```

- 然后直接输入 \*，即可批量添加到每行的行首：



```
35  
36  
37  
38 * 举办市场活动期间，往往  
39  
40  
41  
42  
43  
44  
45  
46
```



# 编辑器通用功能和逻辑

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:11:05

# IDE通用功能和逻辑

下面整理IDE的通用逻辑和功能：

## 动态提示=自动完成=代码补全

一般来说，指的是IDE集成的动态提示的功能

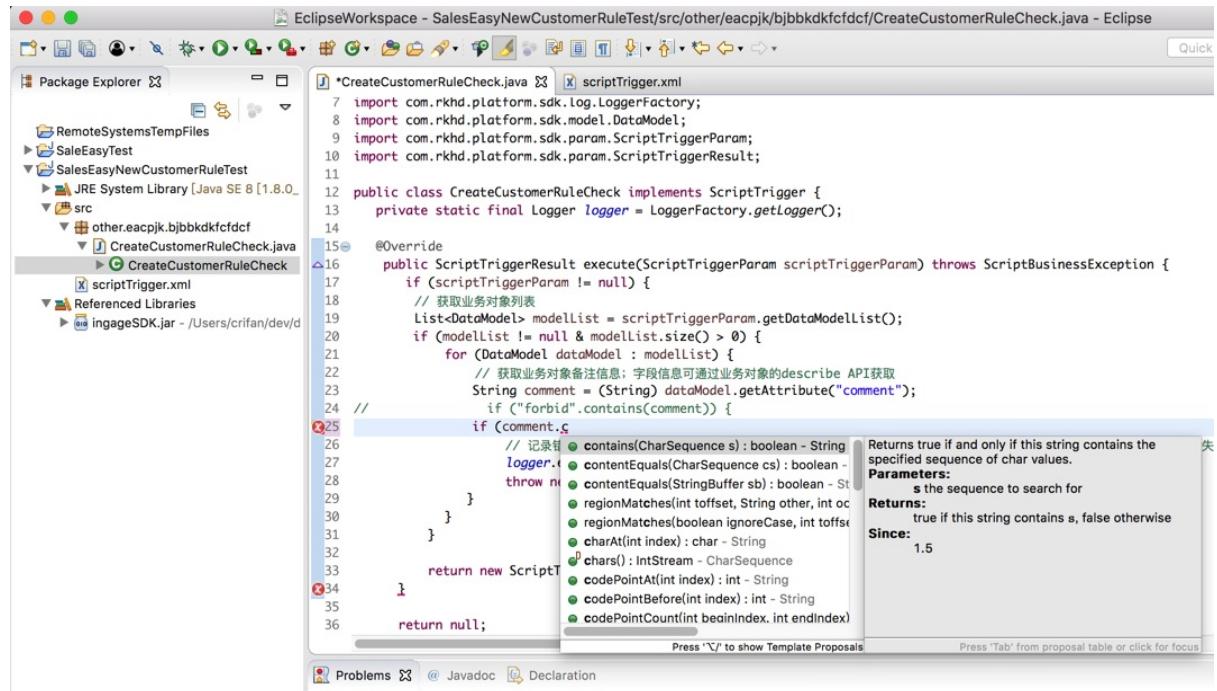
当前，现在有些功能强大的编辑器，比如VSCode，也支持部分语言（或库）的动态提示了。

注：VSCode中的动态提示，叫做：IntelliSense

举例：

### Eclipse的动态补全提示

比如字符串支持哪些c开头的函数或属性：



### PyCharm的动态补全提示

比如Selenium的driver有哪些函数或属性：

A screenshot of the PyCharm IDE interface. The code editor shows Python code related to Selenium WebDriver. A tooltip is open over the method `find\_element\_by\_xpath`, displaying its documentation and several other methods from the WebDriver class. The methods listed are: find\_element\_by\_xpath, find\_element, add\_cookie, application\_cache, back, capabilities, close, command\_executor, create\_options, create\_web\_element, current\_url, current\_window\_handle, delete\_all\_cookies, delete\_cookie, desired\_capabilities, error\_handler, execute, and execute\_async\_script. Each method is preceded by a color-coded icon (method/magic, property, function) and its return type (WebDriver). The tooltip also includes a note at the bottom: "Press ^ to choose the selected (or first) suggestion and insert a dot afterwards >>".

```

# selenium driver
# driver = None
driver = webdriver.Chrome()
driver.

gCfg m find_element_by_xpath(self, xpath) WebDriver
m find_element(self, by, value) WebDriver
m add_cookie(self, cookie_dict) WebDriver
p application_cache WebDriver
m back(self) WebDriver
f capabilities WebDriver
m close(self) WebDriver
f command_executor WebDriver
m create_options(self) WebDriver
m create_web_element(self, element_id) WebDriver
p current_url WebDriver
p current_window_handle WebDriver
m delete_all_cookies(self) WebDriver
m delete_cookie(self, name) WebDriver
p desired_capabilities WebDriver
f error_handler WebDriver
m execute(self, driver_command, params) WebDriver
m execute_async_script(self, script, args) WebDriver

```

Press ^ to choose the selected (or first) suggestion and insert a dot afterwards >>

## Xcode的动态补全提示

XCGLogger的log的属性level有哪些

A screenshot of the Xcode IDE interface. The code editor shows Swift code where a variable `logLevel` is being assigned a value. A tooltip is open over the assignment, showing the available levels: all, debug, error, info, init(rawValue: Int), none, RawValue, and severe. The tooltip also includes a note: "M Int RawValue".

```

func initLog() {
    gLog = XCGLogger.default//XCGLogger.defaultInstance()

    //var logLevel:XCGLogger.LogLevel = XCGLogger.LogLevel.debug

    var logLevel:XCGLogger.Level = XCGLogger.Level.
    if LogLevelIsVerbose { [XCGLogger.Level] all
        logLevel = XCGLogger.
    }
    var fileLogLevel:XCGLogger.LogLevel = XCGLogger.LogLevel.debug
    if FileLogLevelIsVerbose { XCGLogger.Level debug
        fileLogLevel = XCGLogger.
    }
    gLog.setup(
        level: logLevel,
        showThreadName: true,
        showLevel: true,
        showFileNames: true,
        showLineNumbers: true,
        writeToFile:LogFile,
        fileLevel: fileLogLevel)
}

```

## VSCode的动态补全提示

VSCode, 在安装了补全插件后:

代码调试期间, 看看python的os模块的有哪些函数:

The screenshot shows a Python code editor with the following code:

```

02
63     ... prevFilenameFiltered = re.sub(r"\\", "/", prevFilePath)
64     logging.debug("prevFilenameFiltered=%s", prevFilenameFiltered)
65     prevFullFilename = os.path..join(filePathPrefix, prevFilenameFiltered)
66
67     ⚡ getatime
68     ⚡ getctime
69     ⚡ getmtime
70     ⚡ getSize
71     ⚡ isabs
72     ⚡ isdir
73     ⚡ isfile
74     ⚡ islink
75     ⚡ ismount
76     ⚡ join
77     ⚡ lexists
78     ⚡ normcase
    ... splittedPath = unifiedFilePath.split("/")

```

A tooltip for the `join` method is displayed, showing its documentation:

`def join(path, *paths)`  
Join two or more pathname components, inserting "" as needed.

## 焦点提示

在写完代码时，鼠标移动上去还可以看到函数/变量的描述和提示

算是自动完成的相关功能

### Eclipse的焦点提示

提示Java的string的contains的函数功能说明：

The screenshot shows an Eclipse IDE window with Java code. A tooltip for the `contains` method is open:

`boolean java.lang.String.contains(CharSequence s)`

**Parameters:**  
`s` the sequence to search for

**Returns:**  
true if this string contains `s`, false otherwise

**Since:**  
1.5

### VSCode的焦点提示

安装了插件的文本编辑器VSCode中也支持提示：函数/变量/库的说明

提示Python的urllib2的功能介绍：

The screenshot shows a code editor with several tabs at the top: BlogsToWordpress.py, BlogCsdn.py, crifanLib.py (which is the active tab), and cmd\_example.txt.

The code in crifanLib.py is as follows:

```

1209     return ISDOWNOK;
1210
1211     #
1212 def getUrlResponse(url, postDict={}, headerDict={}, timeout=0, useGzip=False, postDataDelimiter="&") :
1213     """Get response from url, support optional postDict,headerDict,timeout,useGzip
1214
1215     Note:
1216     1. if postDict not null, url request auto become to POST instead of default GET
1217     2. if you want to auto handle cookies, should call initAutoHandleCookies() before use this function.
1218         then following urllib2.Request will auto handle cookies
1219     """
1220
1221     # makesure url is string, not unicode, otherwise urllib2.urlopen will error
1222     url = str(url);
1223
1224     if (postDi
1225         if(pos
1226             po
1227             else:
1228                 po
1229                 fo
1230
1231             postDa
1232             #loggi
1233             req =
1234             #loggi
1235             req.ad
1236             else :
1237                 req = urllib2.Request(url);
1238
1239             defHeaderDict = {
1240                 'User-Agent' : gConst['UserAgent'],
1241                 'Cache-Control' : 'no-cache',
1242                 'Accept' : '*/*',
1243                 'Connection' : 'Keep-Alive',
1244             };
1245
1246
1247     realContentRowStartNum = 0
1248
1249     if type == StorybookResourceTy
1250         wsIn = wsInStorybook
1251         wsOut = wsOutStorybook
1252         realContentRowStartNum = 3
1253     else:
1254         wsIn = wsInSong
1255         wsOut = wsOutSong
1256         realContentRowStartNum = 2
1257
1258     for wsInCurRowNum, eachRow in enumerate(wsIn.iter_rows(min_row=realContentRowStartNum)):
1259         wsInCurRowNum += realContentRowStartNum
1260         logging.info("-"*30 + " row[%d] " + "-"*30, wsInCurRowNum)
1261
1262
1263
1264
1265
1266
1267
1268
1269

```

Two code completion popups are shown:

- urllib2**: An extensible library for opening URLs using a variety of protocols. The simplest way to use this module is to call the urlopen function, which accepts a string containing a URL or a Request object (described below). It opens the URL and returns the results as file-like object; the returned object has some extra methods described below.
- enumerate**: Return an enumerate object. iterable must be another object that supports iteration. The enumerate object yields pairs containing a count (from start, which defaults to zero) and a value yielded by the iterable argument. enumerate is useful for obtaining an indexed list: `(0, seq[0]), (1, seq[1]), (2, seq[2]), ...`

## Xcode的焦点提示

Xcode 9 中需要使用按钮才能看到函数描述

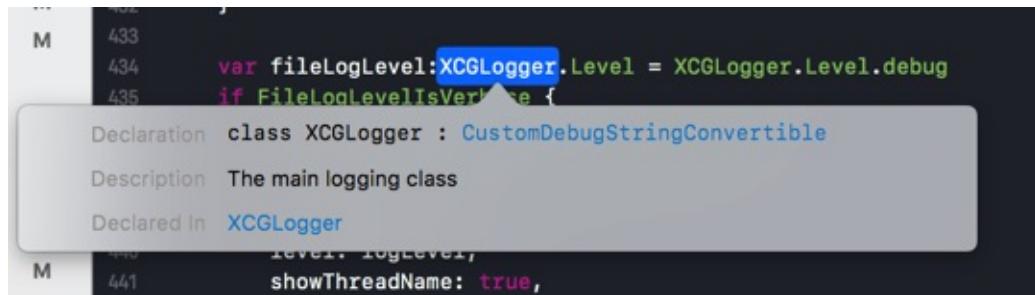
command后，选中：Alt

```

433
434     var fileLogLevel:XCGLogger.Level = XCGLogger.Level.debug
435     if FileLogLevelIsVerbbose {
436         fi
437     }
438
439     gLog.s
440     le
441     sh
442     sh
443     sh
444     showLineNumbers: true,

```

才能看到函数描述:



## 智能检测：没有用到的变量的提示警告

对于代码中，定义了却没用到的变量，会智能检测出来并提示

IDE: Xcode支持

IDE: PyCharm也支持

比如下图中的cmdPara1，没有被用到：

- 颜色：灰色
  - 而普通代码是黑色
    - 能区分出来
- 鼠标移动上去，会提示

◦

## Editor: VSCode后来也支持

VSCode 1.24之后也支持:

Unused variable detection - Unused variables are greyed-out in your JavaScript/TypeScript files

# 调试

## 调试时显示的信息

各种IDE, 调试期间, 所显示的信息, 大致都是这几类:

- 变量
  - 局部变量
  - 全局变量
  - 监视的变量
  - 等等
- 断点
  - 查看全部断点
  - 禁止相关断点
  - 等等

## 调试代码的通用逻辑

### 变量值变化时高亮提示

当你调试代码时, 变量值和之前相比有改变时, 调试工具会标识出来的:

- 举例
  - VSCode
    - 之前

```

35
36     def detectImageInfoFromUrl():
37         for curImgUrl in TestImgUrlList:
38             print("curlImgUrl=%s" % curImgUrl)
39             # imgRawData = getImgRawData(curImgUrl)
40             # img = Image.open(imgRawData)
41
42             getOk, imgFileOrErrMsg = getFile(curImgUrl)
43             print("getOk=%s, imgFileOrErrMsg=%s" % (getOk, imgFileOrErrMsg))
44             if getOk:
45                 imgFile = imgFileOrErrMsg
46                 print("imgfile=%s" % imgFile)
47                 img = Image.open(imgFile)
48                 print("img=%s" % img)
49                 (imgWidth, imgHeight) = img.size
50                 print("imgWidth=%s, imgHeight=%s" % (imgWidth, imgHeight)) # 1x1 / 354x500
51             else:
52                 print("Get image url %s failed: %s" % (curImgUrl, imgFileOrErrMsg))
53
54     if __name__ == "__main__":

```

### ■ 当行运行之后

```

35
36     def detectImageInfoFromUrl():
37         for curImgUrl in TestImgUrlList:
38             print("curlImgUrl=%s" % curImgUrl)
39             # imgRawData = getImgRawData(curImgUrl)
40             # img = Image.open(imgRawData)
41
42             getOk, imgFileOrErrMsg = getFile(curImgUrl)
43             print("getOk=%s, imgFileOrErrMsg=%s" % (getOk, imgFileOrErrMsg))
44             if getOk:
45                 imgFile = imgFileOrErrMsg
46                 print("imgfile=%s" % imgFile)
47                 img = Image.open(imgFile)
48                 print("img=%s" % img)
49                 (imgWidth, imgHeight) = img.size
50                 print("imgWidth=%s, imgHeight=%s" % (imgWidth, imgHeight)) # 1x1 / 354x500
51             else:
52                 print("Get image url %s failed: %s" % (curImgUrl, imgFileOrErrMsg))
53
54     if __name__ == "__main__":

```

- PyCharm

- TODO:

- 添加例子过来

- 值的变化是用红色标识的

## 断点调试

对于断点的逻辑也是通用的：

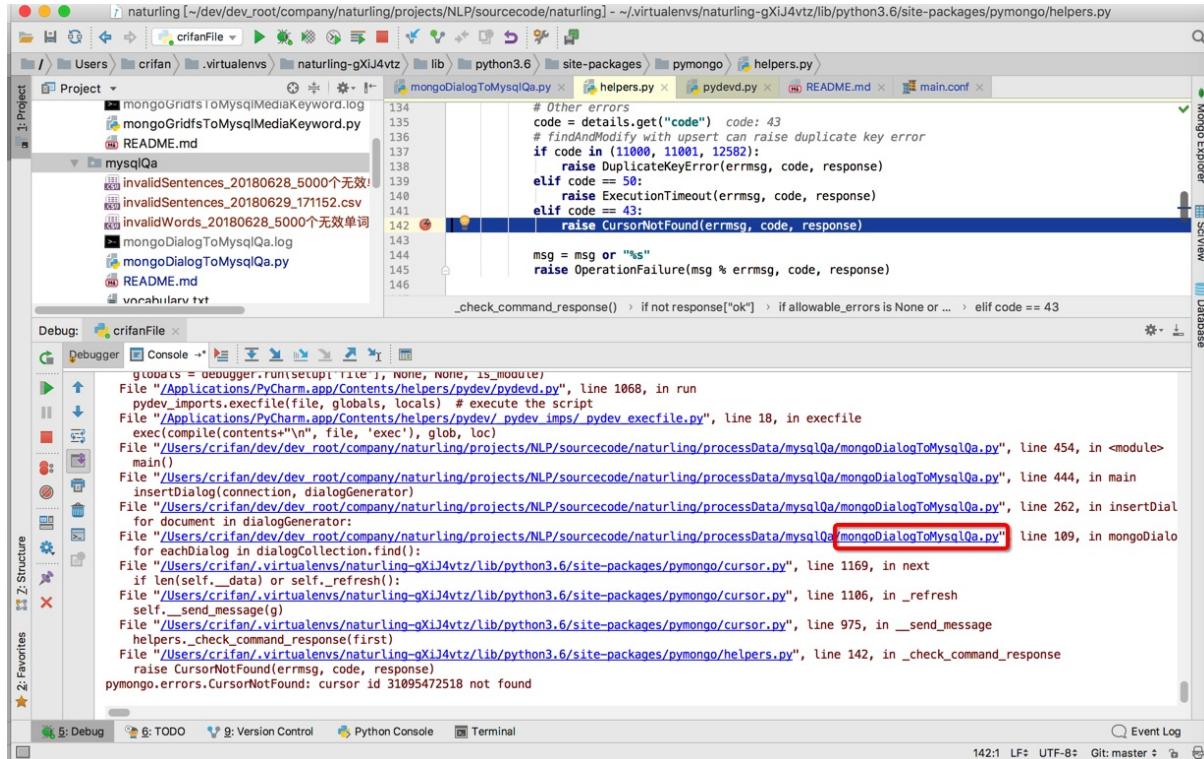
比如运行到 断点 暂停执行时，点击断点的 调用堆栈 call stack，可以快速定位到代码位置：

- PyCharm调试Python

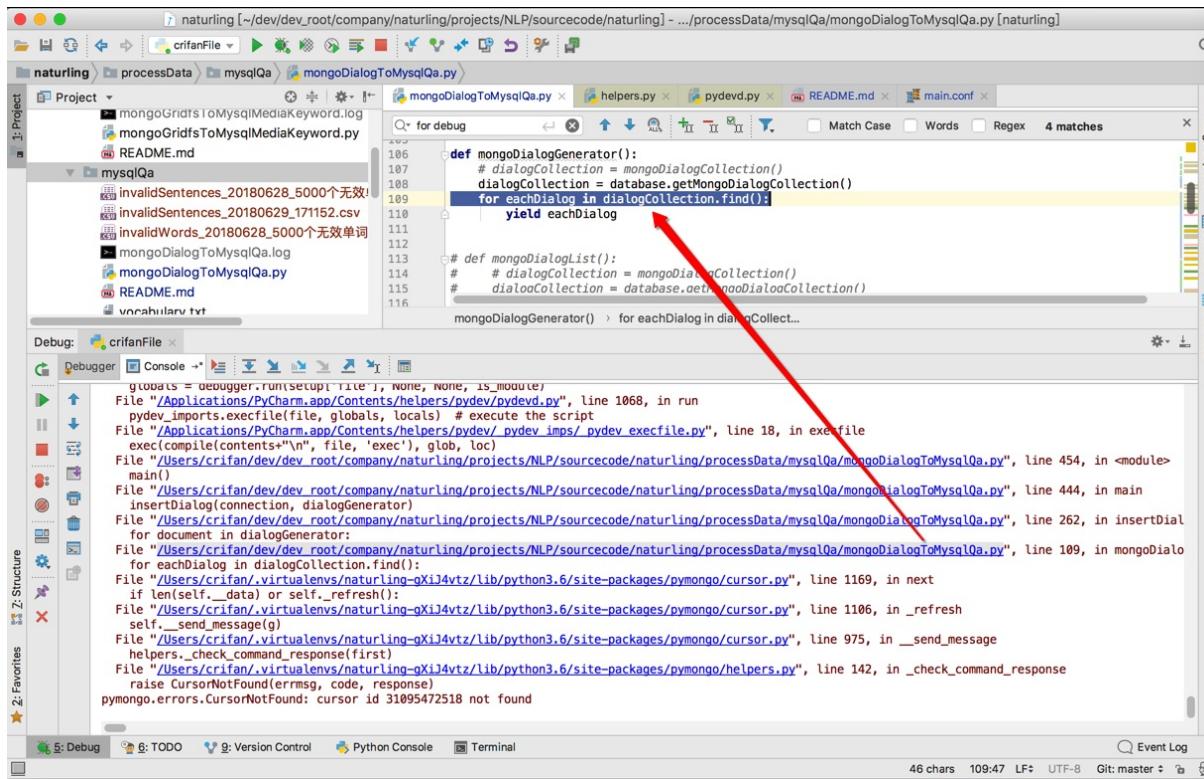
- 
- 点击后，可以跳转到对应代码的位置
  
- 

## 调试异常

调试出错异常时，点击可跳转到出错的代码的位置：



点击对应错误信息中的代码位置，可以跳转到对应源文件：



## 编译

IDE中各种界面上功能内部都是对应的调用底层的命令去实现的

比如IDE中的编译这个按钮，点击后，内部的过程是：

使用不同的编译器，传递对应的include的头文件、查找库函数的路径、链接对应的库等等，最终去编译生成对应的输出的目标文件的

比如：

[已解决] Xcode项目编译出错：Command failed due to signal: Segmentation fault: 11

中而发现：

Xcode 中底层是用 swiftc 去编译 swift 代码的

关于更多的例子，详见：[编译和链接 · 计算机编程通用逻辑知识概念](#)

## 集成终端Terminal

很多IDE集成了终端

好处是在用ide开发期间，同时使用终端去做其他方面的测试。

举例：

- VSCode 写 gitbook 时：用终端去build编译输出为html
- pycharm 时，调试 crifanLib.py 时：测试提取脚本文件名，用终端测试能否正常解析输入的文件名

## 重构-重命名

对于重构中的变量、函数、类等重命名，在逻辑不复杂，且在当前单一文件内，其实多数编辑器也是支持的。

但其实是很多更加智能的IDE，对于重命名，支持的会更好。

比如：

- PyCharm支持重命名
- 新版Xcode9 (+ swift 3.2? / 4.0?) 支持swift的重构，包括rename了
  - 之前版本是不支持swift的rename的

crifan.com，使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:10:59

# 编辑器和IDE详解

注意：此处并不是把市面上存在的所有编辑器和IDE都详细解释一遍，而只是介绍相对主流的或者说在我眼中比较主流和好用的部分编辑器。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12 23:00:34

# 编辑器详解

下面介绍一些常见的主流的编辑器Editor。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:09:57

# VSCode

详见：

[史上最好用的编辑器：VSCode](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:10:02

# Sublime Text

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:10:13

# Atom

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:10:43

# vi/vim

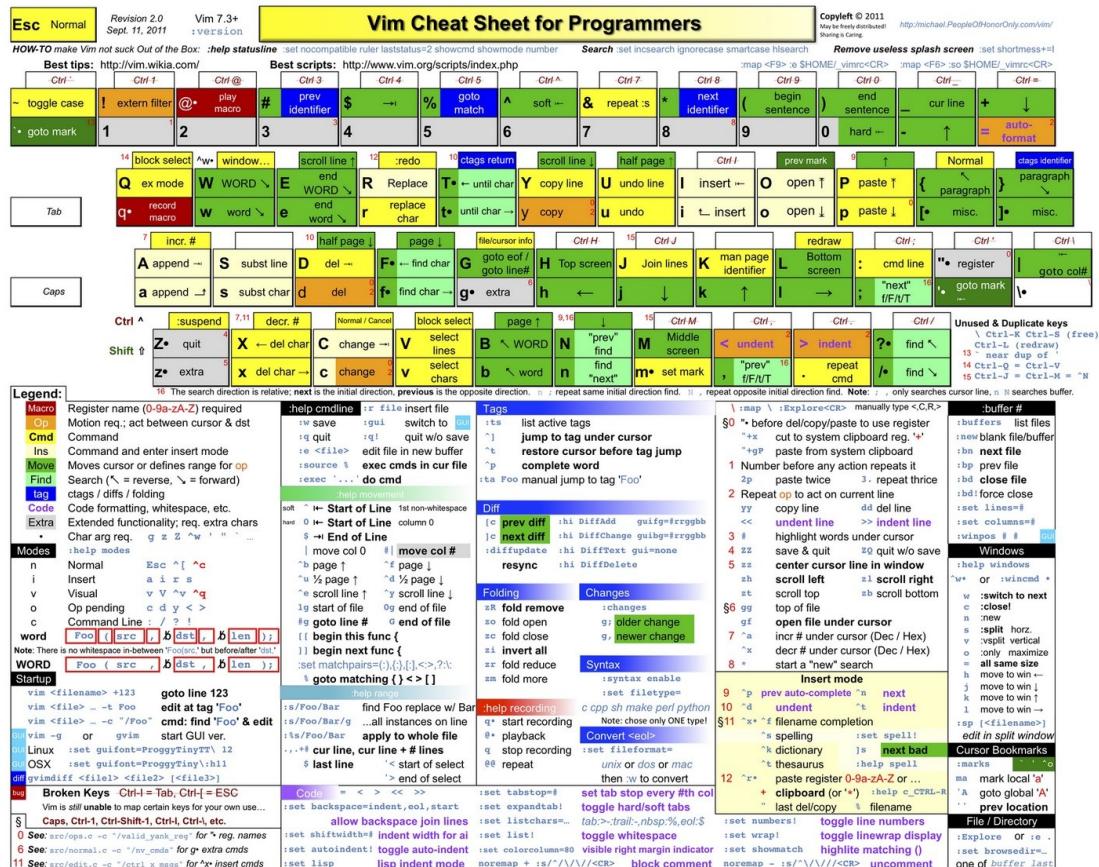
可以去通过这个游戏：

PacVim - A CLI Game To Learn Vim Commands - OSTechNix

去学会常见快捷键操作。

## vi/vim常用快捷键

### vi常用快捷键-键盘图



### vi常用快捷键-表格

| Key                               | What it does                    |
|-----------------------------------|---------------------------------|
| q                                 | quit                            |
| wq                                | write and quit == save and quit |
| q!                                | quit without save changes       |
| Move relative to current position |                                 |

|                                      |   |
|--------------------------------------|---|
| <code>h</code>                       | move left   |
| <code>j</code>                       | move down   |
| <code>k</code>                       | move up   |
| <code>l</code>                       | move right  |
| <b>Move relative to Word</b>         |   |
| <code>w</code>                       | Move to next word   |
| <code>W</code>                       | Move to next blank delimited word                           |
| <code>b</code>                       | Move to the beginning of the word                           |
| <code>B</code>                       | Move to the beginning of blank delimited word               |
| <code>e</code>                       | Move to the end of the word                                 |
| <code>E</code>                       | Move to the end of blank delimited word                     |
| <b>Move relative to Current Line</b> |   |
| <code>^</code>                       | Moves to the first non-blank character in the current line  |
| <code>0</code> or <code> </code>     | Move to the begining of the line                            |
| <code>\$</code>                      | Move to the end of the line                                 |
| <code>+</code>                       | Moves to the first character in the next line               |
| <code>-</code>                       | Moves to the first non-blank character in the previous line |
| <b>Move relative to Line of File</b> |   |
| <code>gg</code> OR <code>1G</code>   | Move to the first line of the file                          |
| <code>nG</code> OR <code>:n</code>   | Move to nth line of the file                                |
| <code>G</code>                       | Move to the last line of the file                           |

一些不错的整理：[Vim cheatsheet](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-11  
18:18:08

# Emacs

Emacs，作为一个编辑器，历史悠久，也有很多粉丝。

有些高手，可以把Emacs功能扩展强大，甚至成为一个操作系统。

不过我个人基本没用过。

- 截图举例：

◦

- 想要学习和了解的，可以参考：
  - [一年成为Emacs高手 像神一样使用编辑器](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12 23:10:39

## Notepad++

详见：

[【crifan推荐】轻量级文本编辑器，Notepad最佳替代品：Notepad++](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:10:29

# Source Insight

- 功能：主要用于查看 C 语言等代码
  - 比如 Linux 内核源码
- 举例：
  - 查看嵌入式C语言的代码：

```

PP_RESULT mypp_init(void *pp_hdl, SYS_INFO *sys_info) {
    MYPP_HANDLER *hdl = pp_hdl;
    char* buf = (char*)pp_hdl + sizeof(MYPP_HANDLER);
    int hdl_size = 0;
    int frame = sys_info->ppc_frame_size;
    PP_RESULT result = SUCCESS;

    print_log("mypp_init!\n");

    // memory handle initialize
    if (mypp_get_mem_size(&hdl_size, sys_info) != SUCCESS) {
        print_must("[mypp_init] get mem size fail\n");
        result = FAIL;
        goto exit;
    }
    hdl_size -= sizeof(MYPP_HANDLER);

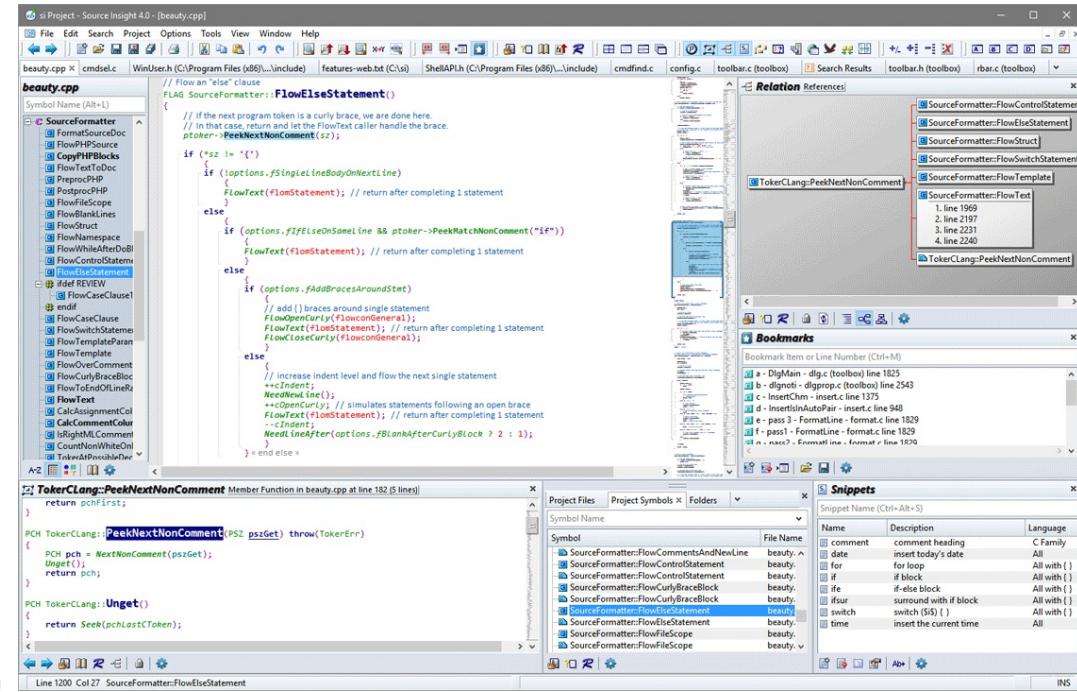
    if (hdl_size < sizeof(int)*frame*TOTAL_CH_NUM) {
        print_must("[mypp_init] fail to allocate memory!\n");
        result = FAIL;
        goto exit;
    }
    hdl->in_buf = (int*)buf;
    buf += sizeof(int)*frame*TOTAL_CH_NUM;
    hdl_size -= sizeof(int)*frame*TOTAL_CH_NUM;

    .....

    shift_init(hdl->obj);

exit:
    return result;
} ? end mypp_init ?

```



crifan.com, 使用署名4.0国际(CC BY 4.0)协议发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:10:26

# IDE详解

此处对于一些主流的好用的IDE分别进行详细介绍。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:01:00

## JetBrains公司

一个专注于各种语言的IDE的公司，旗下有各种很牛的，很智能的IDE：

- for Java : IntelliJ IDEA
- for Python : PyCharm
- for PHP : PhpStorm
- for Web领域开发: WebStorm
- 等等

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:09:12

# PyCharm

详见：

[【整理Book】最智能的Python的IDE：PyCharm](#)

## 优点

- 对于调试支持的很好
  - 比如：
    - [【已解决】Mac中PyCharm中去加断点实时调试scrapy的项目](#)
- 部署和同步也方便
  - 支持自动上传代码实现快速部署
    - [【已解决】PyCharm自动上传改动更新后的文件到CentOS服务器上](#)

## 缺点

其中也有些细节功能不够完美，比如：

- [【未解决】PyCharm中markdown编辑器支持粘贴图片](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12 22:53:59

# IntelliJ IDEA

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:09:26

# PhpStorm

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:09:29

# WebStorm

## 优点

Webstorm中支持了很多第三方工具：



比如管理bug的：

- JIRA
- FogBugz
- Mantis

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:09:16

# Visual Studio

微软的重量级的IDE: `visual Studio`

简称 `vs`

主要用来开发Windows平台的各种语言的程序，典型的有：

- C#
  - 桌面端程序
- F#
- 等等

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12 23:05:52

# Eclipse

一个主流的IDE，是java语言实现的。

特点是支持插件，实现各种功能，支持调试其他各种语言。

- [Eclipse + PyDev](#) : 用于Python开发
  - 比如:
    - [【教程】在 Eclipse 中使用 PyDev 进行 Python 开发 – 在路上](#)

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12 23:06:40

# Android Studio

用于开发安卓Android的app的IDE。

是基于 JetBrains 的 IntelliJ IDEA 改造而来的。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12 23:09:17

# Aptana Studio

Web端开发的一个IDE。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12  
23:07:01

# Xcode

苹果的平台的软件的IDE： Xcode

可用来开发苹果旗下的众多平台的软件：

- 桌面端
  - macOS
- 移动端
  - iOS
  - watchOS
  - tvOS

## 优点

集成了和自己的语言和环境高度整合的功能和智能提示

比如：

```

13     case singleSelect //单选
14     case multiSelect //多选
15   }
16
17   class DealerListViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {
18     var curType:DealerListVCType
19     var listTableView:UITableView
20
21     var allSelectButton:UIButton
22
23     var isSingleSelect:Bool
24     var dealerList = [DealerItem]()
25     var dealerListCompletionHandler:([_ curDealerList:[Dealer]]) -> Void?
26     var dealerCodeList:[String]
27
28     init(curType:DealerlistVCType = .singleSelect,dealerCodeList:[String] = [String]()
29           ,dealerListCompletionHandler:([_ curDealerList:[Dealer]]) -> Void?) = nil{
30       self.curType = curType
31       self.listTableView = UITableView() //-[UIView init] must be used from main thread only
32       self.allSelectButton = UIButton()
33       self.dealerListCompletionHandler = dealerListCompletionHandler
34       self.isSingleSelect = true
35       self.dealerCodeList = dealerCodeList
36       super.init(nibName: nil, bundle: nil)
37
38       // getDealerList()
39       if (self.curType == .multiSelect){
40         self.isSingleSelect = false
41
42         if self.dealerListCompletionHandler != nil{
43           if gInformationStorage.dealerList.count == 0{
44             getDealerList(reloadData: true)
45           }else{
46             self.dealerList = [DealerItem]()
47             for item in gInformationStorage.dealerList{
48               let dealer = DealerItem(name: item.name, code: item.code)
49             }
50           }
51         }
52       }
53     }
54
55     override func viewDidLoad() {
56       super.viewDidLoad()
57       self.title = "经销商列表"
58       self.navigationItem.rightBarButtonItem = UIBarButtonItem(title: "完成", style: .done, target: self, action: #selector(self.done))
59     }
60
61     // MARK: - UITableViewDelegate
62     func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
63       return self.dealerList.count
64     }
65
66     func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
67       let cell = UITableViewCell(style: .default, reuseIdentifier: "Cell")
68
69       let dealer = self.dealerList[indexPath.row]
70
71       cell.textLabel?.text = dealer.name
72       cell.detailTextLabel?.text = dealer.code
73
74       return cell
75     }
76
77     // MARK: - UITableViewDataSource
78     func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
79       let dealer = self.dealerList[indexPath.row]
80
81       if self.isSingleSelect{
82         self.allSelectButton.setTitle("全选", for: .normal)
83         self.allSelectButton.setTitle("取消", for: .highlighted)
84         self.allSelectButton.setTitle("取消", for: .selected)
85         self.allSelectButton.setTitle("全选", for: .disabled)
86
87         self.isSingleSelect = false
88         self.dealerListCompletionHandler([dealer])
89       }else{
90         self.allSelectButton.setTitle("全选", for: .normal)
91         self.allSelectButton.setTitle("取消", for: .highlighted)
92         self.allSelectButton.setTitle("取消", for: .selected)
93         self.allSelectButton.setTitle("全选", for: .disabled)
94
95         self.isSingleSelect = true
96         self.dealerList.append(dealer)
97         self.dealerListCompletionHandler(self.dealerList)
98       }
99     }
100
101    // MARK: - IBAction
102    @IBAction func done(_ sender: UIBarButtonItem) {
103      self.dismiss(animated: true, completion: nil)
104    }
105
106    @IBAction func allSelect(_ sender: UIButton) {
107      if self.isSingleSelect{
108        self.allSelectButton.setTitle("全选", for: .normal)
109        self.allSelectButton.setTitle("取消", for: .highlighted)
110        self.allSelectButton.setTitle("取消", for: .selected)
111        self.allSelectButton.setTitle("全选", for: .disabled)
112
113        self.isSingleSelect = false
114        self.dealerListCompletionHandler([self.dealerList[0]])
115      }else{
116        self.allSelectButton.setTitle("全选", for: .normal)
117        self.allSelectButton.setTitle("取消", for: .highlighted)
118        self.allSelectButton.setTitle("取消", for: .selected)
119        self.allSelectButton.setTitle("全选", for: .disabled)
120
121        self.isSingleSelect = true
122        self.dealerList.append(self.dealerList[0])
123        self.dealerListCompletionHandler(self.dealerList)
124      }
125    }
126
127    // MARK: - Helper
128    func getDealerList(){
129      let dealerList = DealerItem.getDealerList()
130
131      self.dealerList = dealerList
132
133      self.listTableView.reloadData()
134    }
135
136    func getDealerList(reloadData: Bool){
137      let dealerList = DealerItem.getDealerList()
138
139      self.dealerList = dealerList
140
141      if reloadData{
142        self.listTableView.reloadData()
143      }
144    }
145
146    func DealerItem.getDealerList() -> [DealerItem]{
147      let dealerList = DealerItem.getDealerList()
148
149      return dealerList
150    }
151  }

```

此时是当程序崩溃了， 提示对应的出错的地方

到底是哪里的代码写错了： 把UI相关的内容， 放在了非主线程中执行

这样才能帮助我们快速定位问题， 改掉错误代码

-> 否则这类 `NSInternalInconsistencyException` 的问题， 需要花费很大精力， 并且很难定位到具体错误位置， 也就很难解决掉问题的。

Xcode 9 (+ Swift 3.2?) 支持 调试时监测出哪些代码 在后台线程操作主线程=UI线程中的元素了

- » 提示会导致崩溃
- » 有些代码并不会立刻崩溃， 但是往往后续遇到时， 会崩溃
- » 而对于这种问题：

在后台线程操作主线程=UI线程中的元素

- 之前版本的Xcode并没有能力监测出来
  - 所以如果需要自己去找，就很麻烦
    - 即使给出了崩溃日志，出错的代码的定位也都很困难
- 同样的有问题的代码，在iOS11之前的，没有崩溃
  - 但是运行到iOS11的机器上，结果就崩溃了
    - 详见：
      - [【已解决】Xcode9编译项目出错：Terminating app due to uncaught exception NSInternalInconsistencyException reason Only run on the main thread](#)

## 查看到即使没文档的库的有哪些函数

对于集成了的第三方库，即使没有文档，也可以通过Xcode的智能提示，而间接知道有哪些函数

比如：

[【已解决】用飞语FYRtcEngineKit去实现基本的iOS间的语音通话](#)

期间，对于飞语的SDK有哪些函数，除了看头文件 `FYRtcEngineKit.h` 之外，还可以通过，在设置了当前类为代理的前提下，输入对应字符串，而动态提示出对应的函数：

The screenshot shows the Xcode interface with a Swift file open. The code completion dropdown is displayed, showing various methods from the `FYRtcEngineKit` class. The methods listed include `onFYRtcEngine(_:engine:callEnd status:)`, `onFYRtcEngineCalleePrepareSucess(_engine:)`, `onFYRtcEngineDialBackSuccess(_engine:)`, and others. The dropdown also includes a note at the bottom: "Event of the DialBack success."

```

func endCallCallback(status:FYRtcEngineStatus?) {
    self.infoNotice("endCallCallback:status=\(String(describing: status))")
}

// 点到点语音相关Delegate 方法(FYRtcEngineKitDelegate)
// -----
// func onFYRtcEngineCalleePrepareSucess(_ engine: FYRtcEngineKit!) {
//     self.infoNotice("onFYRtcEngineCalleePrepareSucess")
// }

fyertd
M onFYRtcEngine(_ engine: FYRtcEngineKit!, callEnd status: FYRtcEngineStatus!)
M onFYRtcEngineCalleePrepareSucess(_ engine: FYRtcEngineKit!)
M onFYRtcEngineDialBackSuccess(_ engine: FYRtcEngineKit!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, outgoingCall callee: String!, uid: String!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, didJoinOfUid uid: String!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, didLeaveOfUid uid: String!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, didJoinChannel channelId: String!, uid: String!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, didLeaveChannelId: String!, stats status: FYRtcEngineStatus!)
M onFYRtcEngine(_ engine: FYRtcEngineKit!, reportRtcStats status: FYRtcEngineStatus!) {
    let logStr = "reportRtcStats: 累计发送字节数:\(status.sendBytes), 累计接收字节数:\(status.receiveBytes)"
}

Event of the DialBack success.

```

另外，智能提示可以（在.swift代码中）帮助（从OC的函数）自动获得Swift版本的函数：

对于官网和头文件都只提供了OC的库函数，比如：

```
- (void)onFYRtcEngine:(FYRtcEngineKit *)engine didAudioMuted:(BOOL)muted uid:(NSString *)uid;
```

而如何转换为此处希望的Swift的版本，则也可以通过智能提示而自动输入：

The screenshot shows the Xcode interface with a Swift file open. The code completion dropdown is displayed, showing various methods from the `FYRtcEngineKit` class. The methods listed include `onFYRtcEngine(_:engine:didAudioMuted:muted:Bool, uid:String!)`, `provideImageData(_data: UnsafeMutableRawPointer, _size width: Int, _height: Int, userInfo info: Any?)`, `childViewControllerForHomeIndicatorAutoHidden() -> UIViewController?`, and others. The dropdown also includes a note at the bottom: "Event of user audio muted or unmuted."

```

79 didaudio|
80 M onFYRtcEngine(_ engine: FYRtcEngineKit!, didAudioMuted muted: Bool, uid: String!)
81 M provideImageData(_ data: UnsafeMutableRawPointer, _size width: Int, _ height: Int, userInfo info: Any?)
82 M childViewControllerForHomeIndicatorAutoHidden() -> UIViewController?
83 M didUpdateFocus(in context: UIFocusUpdateContext, with coordinator: UIFocusAnimationCoordinator)
84 M allowedChildViewControllersForUnwinding(from source: IStoryboardUnwindSegueSource) -> [UIViewController]
85 M observe<Value>(_ keyPath: KeyPath<ViewController, _ rivedChange<Value>) -> Void) -> NSKeyValueObservation
86 M didChange(_ changeKind: NSKeyValueChange, valuesAt indexes: IndexSet, forKey key: String)
87 M didChange<Value>(_ changeKind: NSKeyValueChange, v...exSet, for keyPath: KeyPath<ViewController, Value>)
88
89 Event of user audio muted or unmuted
90

```

回车后即可自动写出函数原型：

```
76      func onFYRtcEngine(_ engine: FYRtcEngineKit!, didAudioMuted muted: Bool, uid: String!) {  
77          code  
78      }  
79  
80  
81  
82
```

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12 23:02:06

# 微信开发者工具

腾讯旗下，用来开发 小程序 的工具。

- 主页：[微信开发者工具](#)
- 截图

◦

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:04:51

# HBuilder

详见：

[记录] Mac中安装和使用前端开发工具：HBuilder

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
23:06:00

## 编辑器和IDE总结

- 总体推荐逻辑：开发不同的技术，推荐用对应合适的编辑器
- 具体推荐哪款：
  - 命令行编辑文件：`vim`
    - 尽量用`vim`, 比`vi`更好用, 往往有代码高亮, 效果更好
  - Android：`Android Studio`
  - Swift/OC：`Xcode`
  - Python
    - 重量级：`PyCharm`
    - 轻量级：`VSCode+Python插件`
  - Web端
    - 小程序：微信开发者工具
    - 也可偶尔配合用`VSCode`编辑代码
    - HTML+CSS+JS
      - 重量级：`WebStorm`
      - 轻量级：`VSCode`
  - 桌面端
    - Windows
      - C#：`Visual Studio`
  - 编辑其他各种文件：`VSCode`

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新：2019-07-12  
22:59:15

## 附录

下面列出相关参考资料。

crifan.com, 使用[署名4.0国际\(CC BY 4.0\)协议](#)发布 all right reserved, powered by Gitbook最后更新: 2019-07-12 23:11:18

# 参考资料

- 史上最好用的编辑器：VSCode
- 【crifan推荐】轻量级文本编辑器，Notepad最佳替代品：Notepad++
- PowerAQ Customized Module Porting Guide \* Zelus 智能语音开发平台
- Vi Cheat Sheet
- A quick reference list of vi editor commands
- vi command summary
- 31 Shortcuts for vi (Linux)
- vim 常用快捷键及使用技巧 - 简书
- PacVim - A CLI Game To Learn Vim Commands - OSTechNix
- Vim cheatsheet
- What a wonderful world ! - Coding~
- 软件技术开发通用知识
- GNU Emacs - GNU Project
- 技术人生的职场众生相 - 十多年的经验与心得 - 灵感之源 - 博客园
- 微软即将推出 Web 版 VS Code (Visual Studio Online) - OSCHINA
- 盘点 2017 年 Python 领域值得关注的 5 个库、工具和开发者
- 码云 Gitee IDE 全新上线 – 码云 Gitee 官方博客
- Visual Studio 与 Eclipse, 谁是最强 IDE?
- [整理] 好用的前端开发工具
- An overview of Visual Studio Code for front-end developers
- Brackets - A modern, open source code editor that understands web design.
- 字符编码简明教程 – 在路上
- 字符编码详解
- [已解决] 企业版iOS的ipa通过OTA发布后还是无法下载和安装
- 【记录】用VSCode开发和调试Python
- 【已解决】VSCode中调试Python代码
- VS Code 列编辑功能说明 - 程序园
- VS Code 列编辑功能说明 - 天马3798-IT大道
- 怎样在 VSCode 中进行列选择? - 知乎
- [已解决] Xcode项目编译出错：Command failed due to signal: Segmentation fault: 11
- 【已解决】Xcode9编译项目出错：Terminating app due to uncaught exception NSInternalInconsistencyException  
reason Only run on the main thread
- 【已解决】用飞语FYRtcEngineKit去实现基本的iOS间的语音通话
- [记录] Mac中安装和使用前端开发工具：HBuilder
- WebStorm 有哪些过人之处？ - 知乎
- 【未解决】PyCharm中markdown编辑器支持粘贴图片
- 【已解决】Mac中PyCharm中去加断点实时调试scrapy的项目
- 【已解决】PyCharm自动上传改动更新后的文件到CentOS服务器上
- Top 48 Integrated Developer Environments (IDEs) & Code Editors | ProfitBricks Blog
- Python-Guide-CN/env.rst at master · Prodesire/Python-Guide-CN
- IDE - 维基百科, 自由的百科全书
- 集成开发环境 - 维基百科, 自由的百科全书
- 【记录】使用Python的IDE：PyScripter – 在路上

