设计文档

开发界面

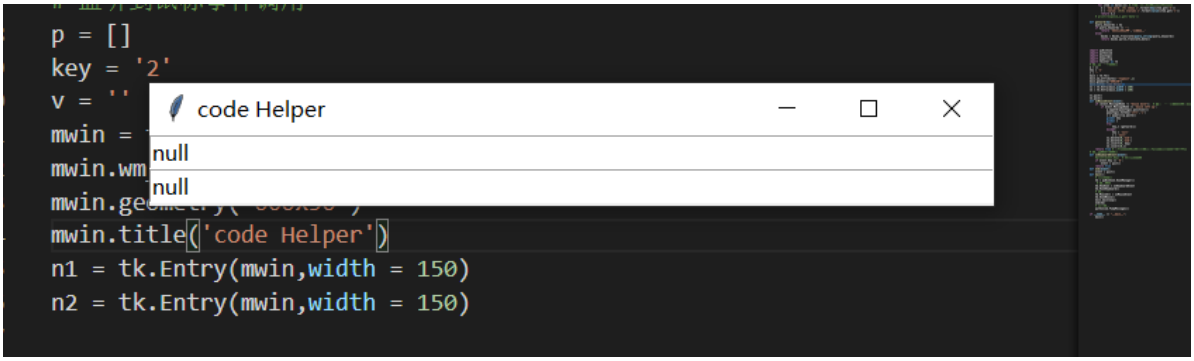
当用户查询的时候提供一个对话框，对话框中返回用户需要的信息，包括文字说明，图片，或是网址来源，因此我们的服务程序是一个pc端的应用程序，程序启动后，监听来自用户的操作，我们使用pyqt5来做应用界面设计。

```
app = QApplication(sys.argv)
app.setApplicationName("Code Helper")
app.setOrganizationName("Code Helper")
```

搜索方式

划词方式

我们首先尝试了使用划词查询的方式，我们知道，市场上有很多很好的划词翻译，只需要鼠标选中再使用快捷键就可以实现查询的功能。于是我们也是实现了一下功能，在第一行搜索关键词，第二行返回搜索得到的html中的文字信息。



实现方式：

```
hm = pyWinhook.HookManager()
    # 监听键盘
    hm.KeyDown = onKeyboardEvent
    hm.HookKeyboard()
    # 监听鼠标
    hm.MouseAll = onMouseEvent
    hm.HookMouse()
    mwin.mainloop()
    end(hm)
    # 循环监听
    pythoncom.PumpMessages()
```

缺点：

1. 返回结果无法进行优化排序，我们也不知道那些结果是最优的那些结果是不好的，这样的话，如果我们检索得到多个结果，不能确定到底需要展示哪一个。
2. 监听本身调用了系统api占用资源大

3. 划词占用了ctrl+c的功能，当真正想复制时，并不能复制到文本信息，影响其他使用。

粘贴方式

如果像有道词典一样，不仅提供划词翻译，还提供一个服务主界面，在主界面可以实现更多高级功能，用户的操作度较高，我们使用了pyqt5提供的搜索工具来实现。

主界面是一个pyqt5的应用程序，我们可以将不知道用法的函数，api,以及报错信息直接复制到主界面，然后搜索，主界面展示所有返回结果，为了给用户提供跟多选择，我们决定即为用户提供相关网站的搜索内容，本身也为用户提供搜索引擎的内容，万一用户觉得有更好的答案。

```
self.urlbar22 = QLineEdit()

navtb2= QToolBar("input")
navtb2.addWidget(self.urlbar22)
```

基于url发现

既然要根据用户粘贴的内容进行搜索就必须要，了解如何把用户内容嵌入url，从而实现对相关资源的检索，我们查找各大开发者常用平台，从中熟悉url与检索的关联，找到匹配检索方式。下面以百度为例：

```
browser.setUrl(QUrl('https://www.baidu.com/s?ie=utf-
8&f=8&rsv_bp=1&tn=baidu&wd='+q))
i = self.tabs.addTab(browser,'baidu')
```

百度搜索方式为在https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=1&tn=baidu&wd=之后+ 查询内容

因此我们只需要把用户输入处理为查询的q字符串拼接即可

智能引流

用户的搜索是多样性的，既有函数用法，又有库的用法，又有报错信息，甚至是想使用一下多线程，可是忘了函数名，我们根据这些输入返回结果，而不是穷尽所有结果。

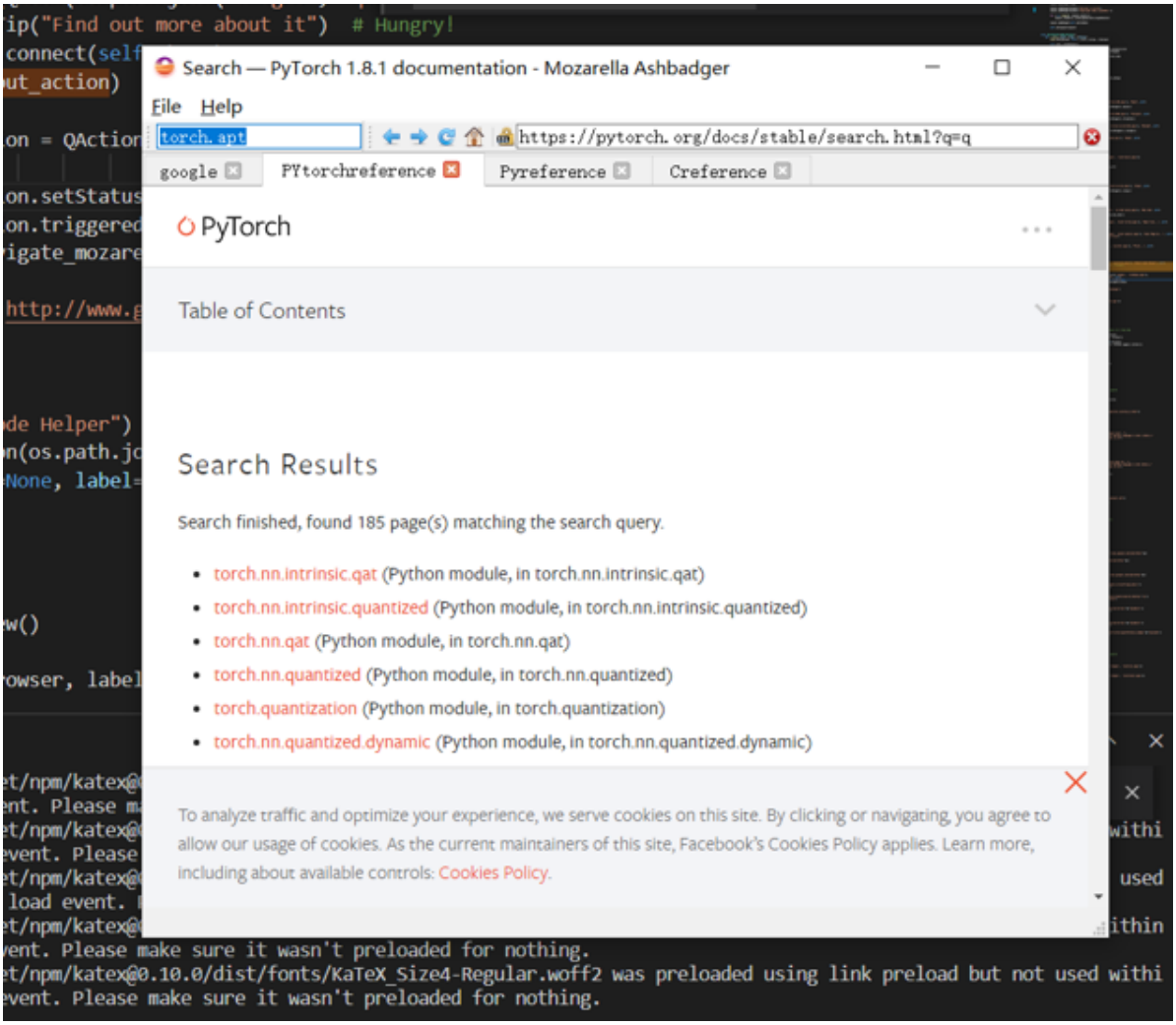
我们提供了以下策略：

1. 如果检测到error存在，判定是用户在debug，转到问题解决类相关资源
2. 如果是其他，匹配到关键词，转换到手册类，如果匹配不到关键词，转换到搜索类相关资源

```
if 'error' in q.lower() or 'warning' in q.lower():

self.tabs.currentWidget().setUrl(QUrl('https://www.google.com/search?q='+q))

            browser = QWebEngineView()
            browser.setUrl(QUrl('https://stackoverflow.com/search?q='+q))
            i = self.tabs.addTab(browser,'debug')
            self.tabs.setCurrentIndex(i)

            browser = QWebEngineView()
            browser.setUrl(QUrl('https://so.csdn.net/so/search?q='+q))
            i = self.tabs.addTab(browser,'csdn')


if 'tf.' in q.lower() :
```

```
                browser = QWebEngineView()

browser.setUrl(QUrl('https://tensorflow.google.cn/swift/api_docs/'))
                i = self.tabs.addTab(browser,'TFreference')
                self.tabs.setCurrentIndex(i)
            if 'torch.' in q.lower():
                browser = QWebEngineView()
                browser.setUrl(QUrl('https://pytorch.org/docs/stable/search.html?
q='+'q'))
                i = self.tabs.addTab(browser,'PYtorchreference')
                self.tabs.setCurrentIndex(i)
            if 'py' in q.lower():
                browser = QWebEngineView()
                browser.setUrl(QUrl('https://www.python.org/search/?
q='+q+'&submit='))
                i = self.tabs.addTab(browser,'Pyreference')
                self.tabs.setCurrentIndex(i)

            else:
            ...
```

如图用户所有torch内置的优化器，检测到关键字后自动引流到pytorch界面



贪婪搜索

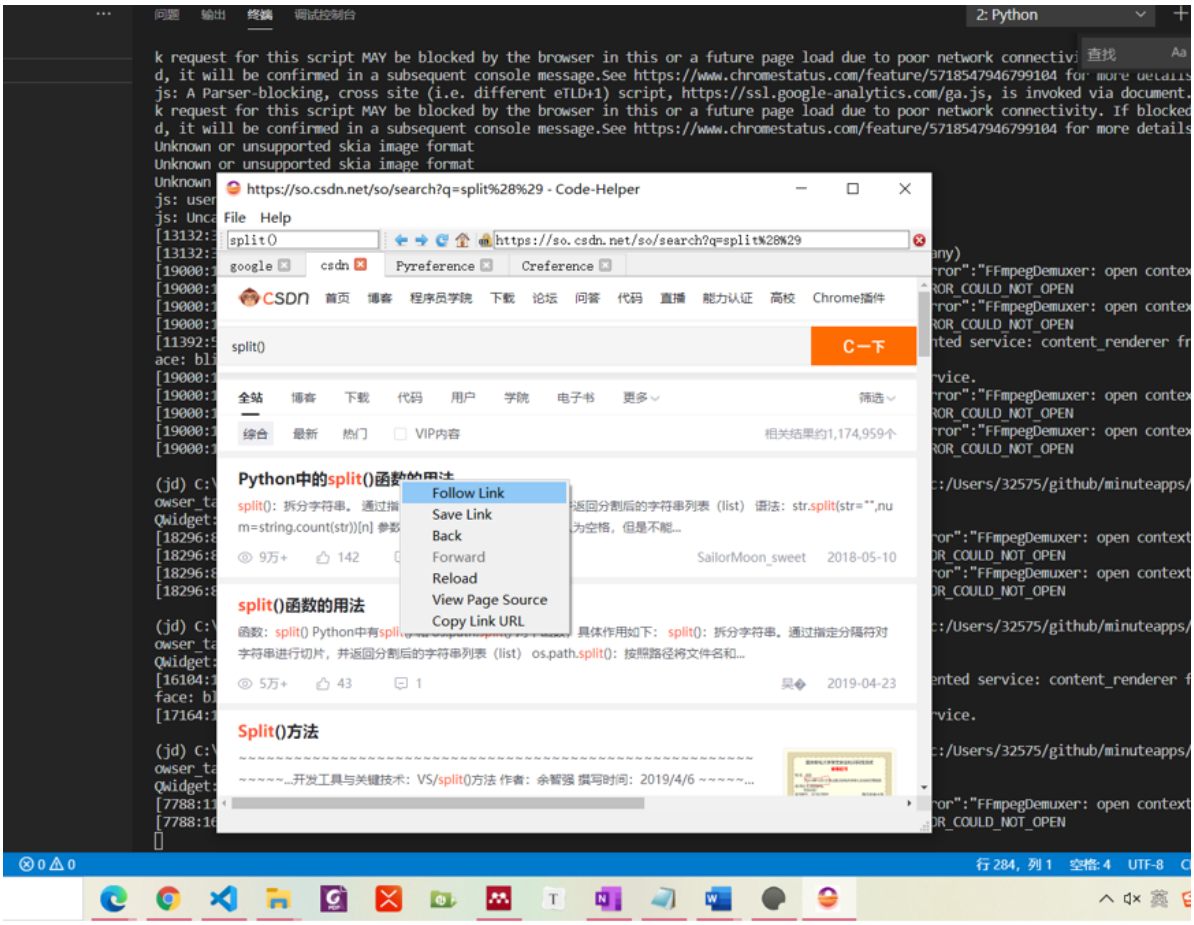
用户常常在看完手册之后不过瘾，可能还想找找有没有源码，之类的，或是用户debug看了一个答案觉得不行，还想看其他答案。这种，都需要我们去提供更多选择。

1. 函数使用查询：我们提供具体**对应语言的手册的搜索结果**
2. bug查询：我们提供stackoverflow，csdn结果

   无论用户是以上那种查询我们都会在提供google和百度的搜索结果，来供用户拓展使用。

```
def add_new_tab(self, qurl=None, label="Blank"):

def add_biying_tab(self, q='https://cn.bing.com/', label="biying"):

self.tabs.currentWidget().setUrl(QUrl("http://www.google.com"))

browser.setUrl(QUrl('https://so.csdn.net/so/search?q='+q))
            i = self.tabs.addTab(browser,'csdn')
```

如果用户不满意提供了几个大型网站的搜索结果，用户可以依据兴趣再次自行检索;如下展示split()搜索用法时的结果。



快捷设计

及时关闭

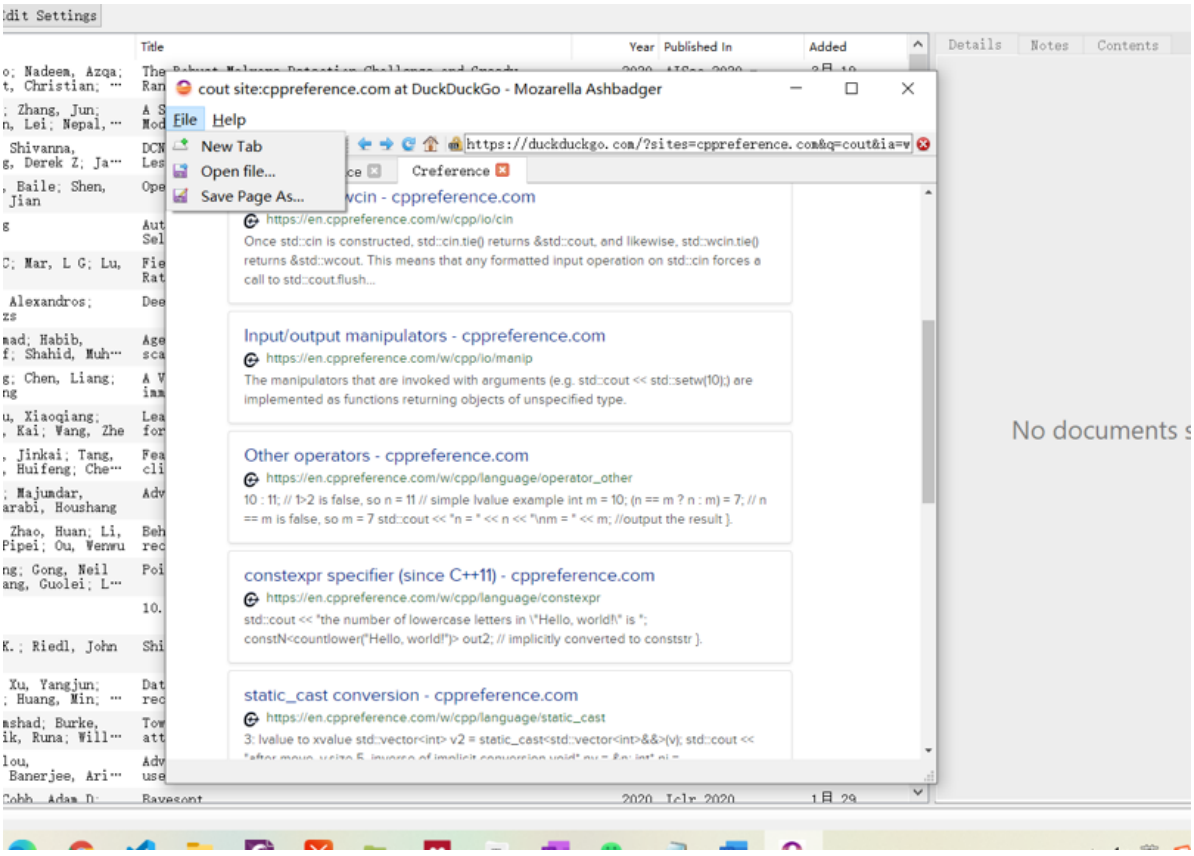
如前面展示，我们的设计面向程序员，尤其是新手程序员，我们不希望写完一段代码后浏览器开了一堆查询各种用法的窗口，我们希望在每次检索的时候，程序能自动帮我们把他们关掉。这样就节约很多人力物力资源。

```
while(1):
            if i==0:
                break
            i -=1
            self.close_current_tab(i)
```

手册服务

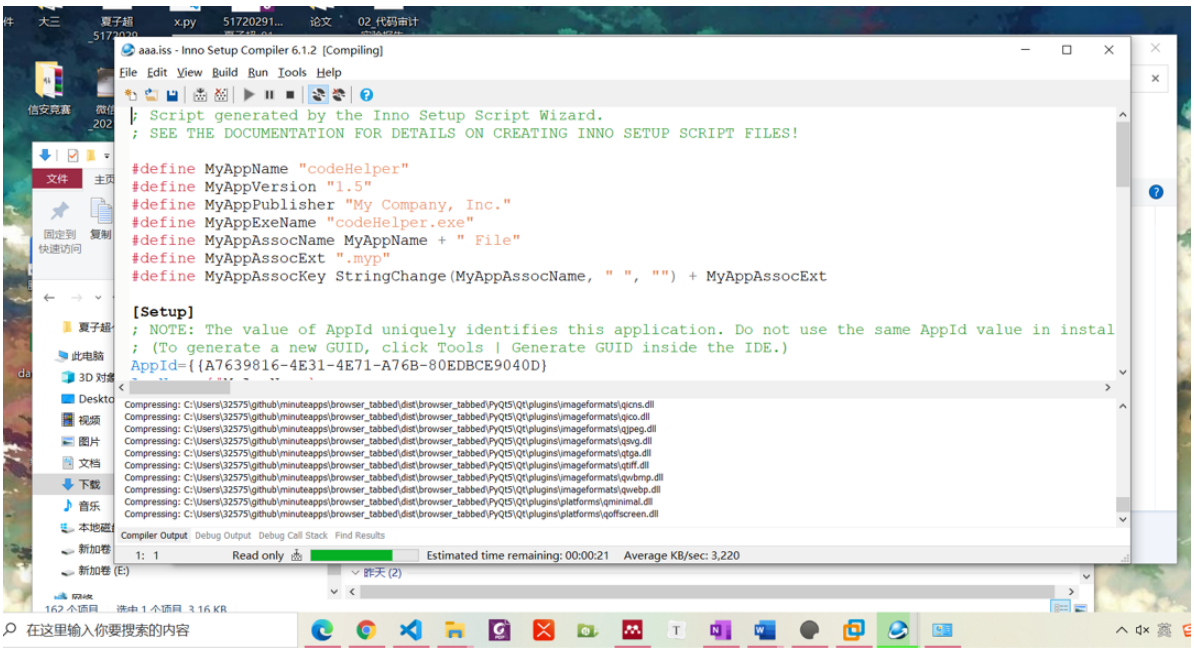
我们提供linux，python, c++等6种手册，用户也可以自己添加手册。

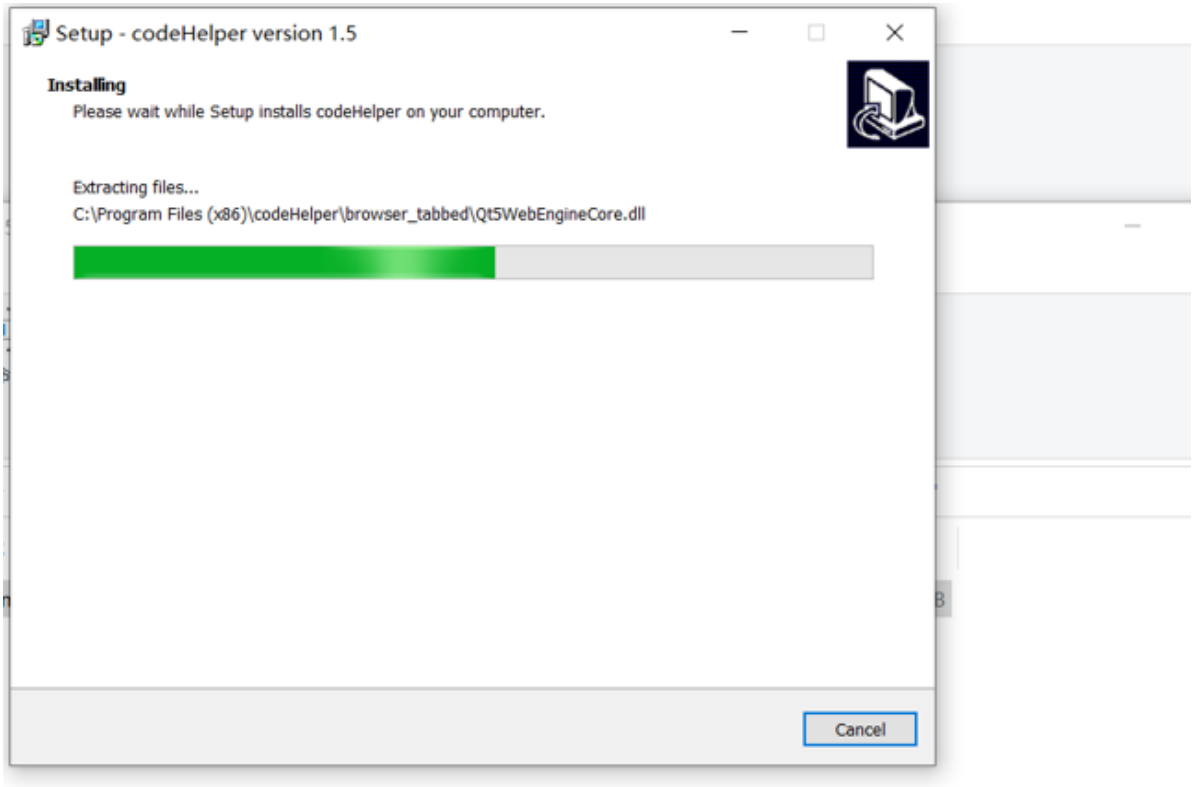
```
navigate_mozarella_action = QAction(QIcon(os.path.join('images',
'lifebuoy.png')),
                                                "Linux handbook", self)
        navigate_mozarella_action.setStatusTip("Go to Linux handbook")
        navigate_mozarella_action.triggered.connect(self.navigate_linux)
```



打包发布

为了提供更加高效的服务，我们希望这个程序发布出来，这样就可以像本机服务程序一样，快捷键启动->搜索，享受更高的生产效率。于是我们采用专用打包工具inno setup，打包后发布。


aaa.iss - Inno Setup Compiler 6.1.2 [Compiling]
File Edit View Build Run Tools Help
; Script generated by the Inno Setup Script Wizard.
; SEE THE DOCUMENTATION FOR DETAILS ON CREATING INNO SETUP SCRIPT FILES!
#define MyAppName "codeHelper"
#define MyAppVersion "1.5"
#define MyAppPublisher "My Company, Inc."
#define MyAppExeName "codeHelper.exe"
#define MyAppAssocName MyAppName + " File"
#define MyAppAssocExt ".myp"
#define MyAppAssocKey StringChange(MyAppAssocName, " ", "") + MyAppAssocExt
[Setup]
; NOTE: The value of AppId uniquely identifies this application. Do not use the same AppId value in instal
; (To generate a new GUID, click Tools | Generate GUID inside the IDE.)
AppId={{A7639816-4E31-4E71-A76B-80EDBCE9040D}
Compiler Output
Debug Output
Debug Call Stack
Find Results
Read only
Estimated time remaining: 00:00:21
Average KB/sec: 3,220


Setup - codeHelper version 1.5
Installing
Please wait while Setup installs codeHelper on your computer.
Extracting files...
C:\Program Files (x86)\codeHelper\browser_tabbed\Qt5WebEngineCore.dll
Cancel