# 跟着鬼哥学PyQt Part.1

<http://blog.csdn.net/guiguzi1110/article/details/39100305>

### 0x1:PyQt的介绍：

PyQt是一个创建GUI应用程序的工具包,它是**[Python](http://lib.csdn.net/base/python" \o "Python知识库" \t "_blank)**编程语言和Qt库的成功融合,Qt库是目前最强大的库之一。PyQt是由Phil Thompson 开发,实现了一个Python模块集。它有超过300类，将近6000个函数和方法,它是一个多平台的工具包，可以运行在所有主要**[操作系统](http://lib.csdn.net/base/operatingsystem" \o "操作系统知识库" \t "_blank)**上，包括UNIX，Windows和Mac。 PyQt采用双许可证，开发人员可以选择GPL和商业许可。在此之前，GPL的版本只能用在Unix上，从PyQt的版本4开始，GPL许可证可用于所有支持的平台。

### 0x2:准备配置环境

我们这里使用的Python版本为2.7.8 ,对应的PyQt版本为4.11.1，下面是这两个的官网：

<https://www.python.org/downloads/>

[http://pyqt.sourceforge.net](http://pyqt.sourceforge.net/)

下面我整理好放到百度网盘的：

<http://pan.baidu.com/s/1dD1o5qT>

两个软件选择32位的，可以运行在64位电脑环境下，win7/win8都已经**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**没问题。

### 0x3:一些学习网站的介绍

[http://www.riverbankcomputing.com](http://www.riverbankcomputing.com/)

<http://wiki.woodpecker.org.cn/moin/PyQt>

<http://www.commandprompt.com/community/pyqt/book1>

<http://wiki.python.org/moin/PyQt>

<http://code.google.com/p/pyqt-doc-cn/>

同时上面百度网盘，我也上传了一本经典的英文资料：

Prentice.Hall.Rapid.GUI.Programming.with.Python.and.Qt.the.definitive.guide.to.PyQt.programming

跟我一样的初学者可以多多读一下这些资料。

### 0x4:编写出来第一个程序

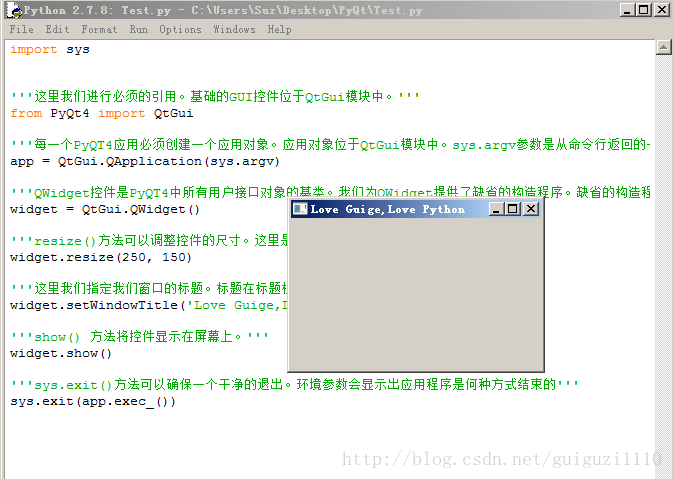
新建一个简单的窗口程序：

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39100305) [copy](http://blog.csdn.net/guiguzi1110/article/details/39100305)

1. **import** sys

4. '''''这里我们进行必须的引用。基础的GUI控件位于QtGui模块中。'''
5. **from** PyQt4 **import** QtGui
7. '''''每一个PyQT4应用必须创建一个应用对象。应用对象位于QtGui模块中。sys.argv参数是从命令行返回的一个参数列表。Python脚本可以从shell来运行。这是一种我们启动脚本的方法。'''
8. app = QtGui.QApplication(sys.argv)
10. '''''QWidget控件是PyQT4中所有用户接口对象的基类。我们为QWidget提供了缺省的构造程序。缺省的构造程序没有父类。一个没有父类的控件叫做窗口。'''
11. widget = QtGui.QWidget()
13. '''''resize()方法可以调整控件的尺寸。这里是250px宽，150px高。'''
14. widget.resize(250, 150)
16. '''''这里我们指定我们窗口的标题。标题在标题栏里显示。'''
17. widget.setWindowTitle('Love Guige,Love Python')
19. '''''show() 方法将控件显示在屏幕上。'''
20. widget.show()
22. '''''sys.exit()方法可以确保一个干净的退出。环境参数会显示出应用程序是何种方式结束的'''
23. sys.exit(app.exec\_())

上面已经写好了注释，比较详细，也是参考一些书籍上面所作出的整理，可以好好理解一下。运行效果如下；



一个简单的界面程序就完成了，那么剩下的我们下一节继续探索添加两个Button，并且让其执行一些功能。

# 跟着鬼哥学PyQtPart.2

<http://blog.csdn.net/guiguzi1110/article/details/39119171>

简述

经过第一篇的简单布局以后，我们来考虑是否可以放上一个按钮，来执行相关的动作呢？答案是肯定可以的，经过**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**，已经简单整理一个按钮的点击事件响应，来执行反编译动作。这里将学习过程记录一下。

### 0x2:思考过程

我们想要做反编译的动作，首先是要准备反编译所需要的相关东西。所以我们下载apktool.jar来执行相关动作，我这里采用的是一个朋友编译的1.5版本的apktool，相关网址为：

<http://bbs.chinaunix.net/thread-4096302-1-1.html>

使用apktool.jar的命令即可，我们可以用**[Python](http://lib.csdn.net/base/python" \o "Python知识库" \t "_blank)**来执行cmd的命令的，所以这个是可以实现的。

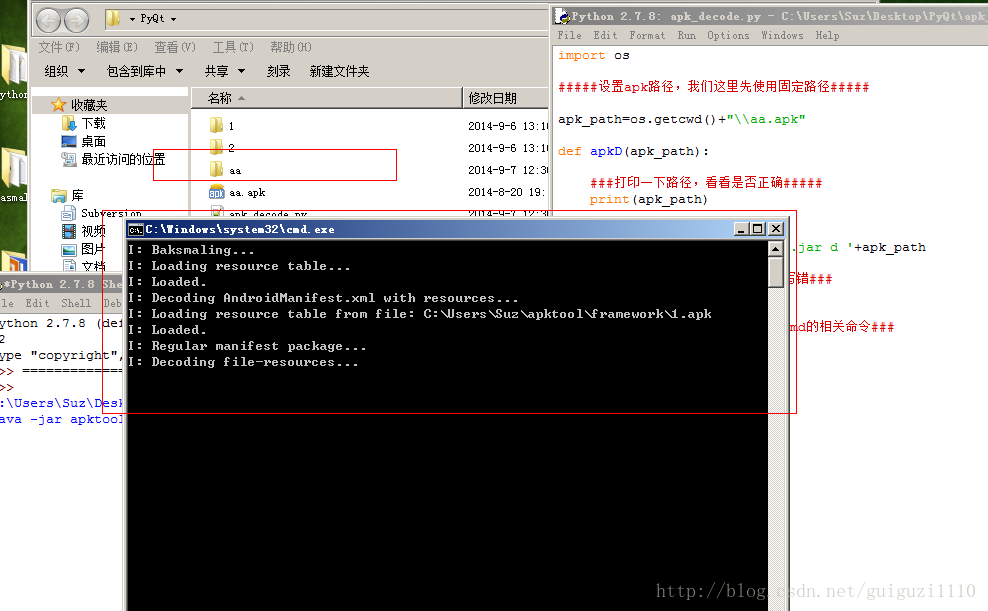
这个方法处理好以后，就是需要跟按钮点击事件关联起来即可。那么下面我们就分开实现这个。

### 0x3:反编译方法的实现

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39119171) [copy](http://blog.csdn.net/guiguzi1110/article/details/39119171)

1. **import** os
3. #####设置apk路径，我们这里先使用固定路径#####
5. apk\_path=os.getcwd()+"\\aa.apk"
7. **def** apkD(apk\_path):
9. ###打印一下路径，看看是否正确#####
10. **print**(apk\_path)
12. ###apktool的命令使用#####
13. apkToD='java -jar apktool.jar d '+apk\_path
15. ###查看一下命令，是否正确，有写错###
16. **print**(apkToD)
18. ###调用os的system命令来执行cmd的相关命令###
19. os.system(apkToD)
21. apkD(apk\_path)

代码中已经写清楚了详细的注释，这里我们可以看一下截图，证明我们的方法是有效的：



我们现在已经可以实现这个功能了，即使用apktool.jar的命令，在命令行下来实现，也是比较简单的，只要配置好[**Java**](http://lib.csdn.net/base/javase)的环境变量即可，那么下一步我们就开始编写一个按钮，实现简单的鼠标点击事件。

### 0x4:鼠标点击事件的实现

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39119171) [copy](http://blog.csdn.net/guiguzi1110/article/details/39119171)

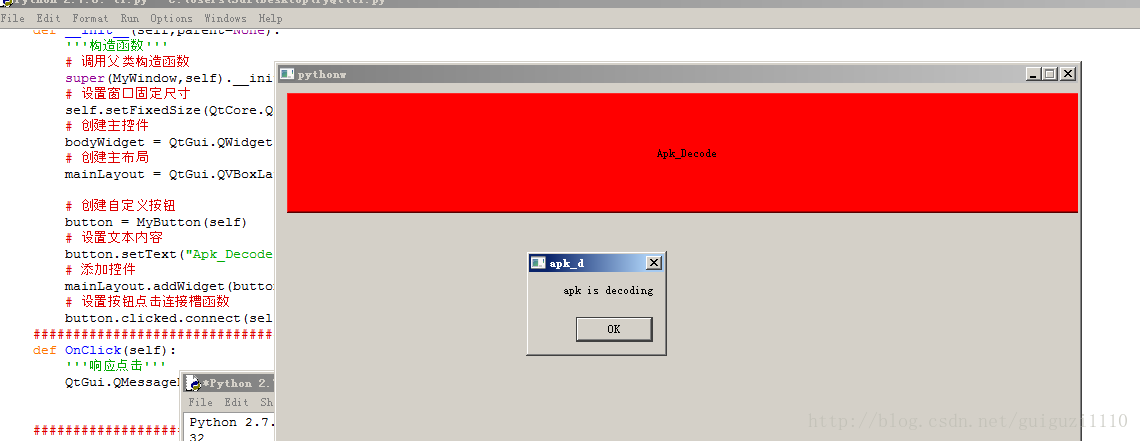
1. **import** sys
2. **import** os
3. **from** PyQt4 **import** QtGui
4. **from** PyQt4 **import** QtCore
6. ###apk的位置，当前类下，我们这里先放到当前文件所在目录###
7. apk\_path=os.getcwd()+"\\aa.apk"
9. ################################# 自定义窗口类 #####################################
10. **class** MyWindow(QtGui.QWidget):
11. '''''自定义窗口类'''
12. ############################# 构造、析构函数 #################################
13. **def** \_\_init\_\_(self,parent=None):
14. '''''构造函数'''
15. # 调用父类构造函数
16. super(MyWindow,self).\_\_init\_\_(parent)
17. # 设置窗口固定尺寸
18. self.setFixedSize(QtCore.QSize(800,600))
19. # 创建主控件
20. bodyWidget = QtGui.QWidget(self)
21. # 创建主布局
22. mainLayout = QtGui.QVBoxLayout(bodyWidget)
24. # 创建自定义按钮
25. button = MyButton(self)
26. # 设置文本内容
27. button.setText("Apk\_Decode")
28. # 添加控件
29. mainLayout.addWidget(button)
30. # 设置按钮点击连接槽函数
31. button.clicked.connect(self.OnClick)
32. ############################### 命令 ########################################
33. **def** OnClick(self):
34. '''''响应点击'''
35. QtGui.QMessageBox.about(self,"apk\_d","apk is decoding")

38. ################################ 事件 ########################################
39. **def** mousePressEvent(self,event):
40. '''''鼠标按下事件'''
41. # 判断是否为鼠标左键按下
42. **if** event.button() == QtCore.Qt.LeftButton:
43. # 设置窗口背景颜色
44. self.setStyleSheet('''''''')




50. ################################## 自定义按钮类 ####################################
51. **class** MyButton(QtGui.QToolButton):
52. '''''自定义按钮类'''
53. ############################## 构造、析构函数 ##################################
54. **def** \_\_init\_\_(self,parent=None):
55. '''''构造函数'''
56. # 调用父类构造函数
57. super(MyButton,self).\_\_init\_\_(parent)
58. # 设置按钮尺寸
59. self.setFixedSize(QtCore.QSize(800,120))
60. # 设置按钮背景颜色
61. self.setStyleSheet('''''background-color:red;''')
62. ################################ 事件 #########################################
63. **def** mousePressEvent(self,event):
64. '''''鼠标按下事件'''
65. # 判断是否为鼠标左键按下
66. **if** event.button() == QtCore.Qt.LeftButton:
67. # 发射点击信号
68. self.clicked.emit(True)
69. # 传递至父窗口响应鼠标按下事件
70. self.parent().mousePressEvent(event)
72. ################################## 主函数 ##########################################
73. **if** \_\_name\_\_ == "\_\_main\_\_":
74. '''''主函数'''
75. # 声明变量
76. app = QtGui.QApplication(sys.argv)
77. # 创建窗口
78. window = MyWindow()
79. # 设置窗口显示
80. window.show()
81. #应用程序事件循环
82. sys.exit(app.exec\_())

代码中有详细的注释，这里是效果截图：



点击apk\_decode按钮，会弹出一个对话框提示正在反编译，说明我们已经将按钮的点击事件写上了，然后下一步我们就是将反编译的方法加上，并且对应放到按钮的点击事件里面。

### 0x5:整合两处代码

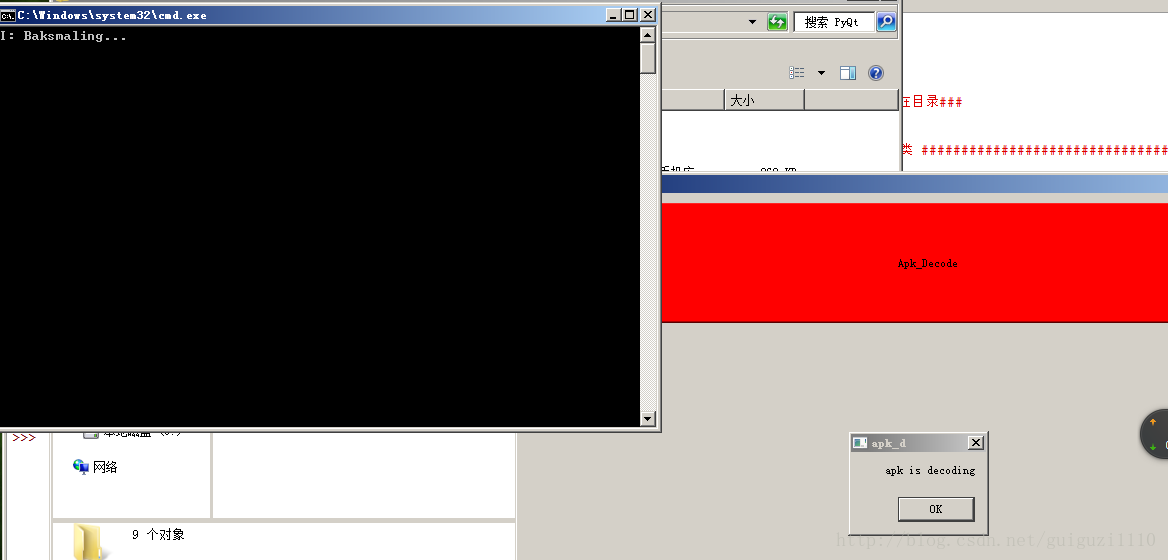
详细代码如下：

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39119171) [copy](http://blog.csdn.net/guiguzi1110/article/details/39119171)

1. **import** sys
2. **import** os
3. **from** PyQt4 **import** QtGui
4. **from** PyQt4 **import** QtCore
6. ###apk的位置，当前类下，我们这里先放到当前文件所在目录###
7. apk\_path=os.getcwd()+"\\aa.apk"
9. ################################# 自定义窗口类 #####################################
10. **class** MyWindow(QtGui.QWidget):
11. '''''自定义窗口类'''
12. ############################# 构造、析构函数 #################################
13. **def** \_\_init\_\_(self,parent=None):
14. '''''构造函数'''
15. # 调用父类构造函数
16. super(MyWindow,self).\_\_init\_\_(parent)
17. # 设置窗口固定尺寸
18. self.setFixedSize(QtCore.QSize(800,600))
19. # 创建主控件
20. bodyWidget = QtGui.QWidget(self)
21. # 创建主布局
22. mainLayout = QtGui.QVBoxLayout(bodyWidget)
24. # 创建自定义按钮
25. button = MyButton(self)
26. # 设置文本内容
27. button.setText("Apk\_Decode")
28. # 添加控件
29. mainLayout.addWidget(button)
30. # 设置按钮点击连接槽函数
31. button.clicked.connect(self.OnClick)
32. ############################### 命令 ########################################
33. **def** OnClick(self):
34. '''''响应点击'''
35. QtGui.QMessageBox.about(self,"apk\_d","apk is decoding")
37. ############################### 反编译方法 ########################################
38. **def** apk\_decode(self,path):
39. **print**(path)
40. apkToD='java -jar apktool.jar d '+path
41. **print**(apkToD)
42. os.system(apkToD)
44. ################################ 事件 ########################################
45. **def** mousePressEvent(self,event):
46. '''''鼠标按下事件'''
47. # 判断是否为鼠标左键按下
48. **if** event.button() == QtCore.Qt.LeftButton:
49. # 设置窗口背景颜色
50. self.setStyleSheet('''''''')
51. self.apk\_decode(apk\_path)



56. ################################## 自定义按钮类 ####################################
57. **class** MyButton(QtGui.QToolButton):
58. '''''自定义按钮类'''
59. ############################## 构造、析构函数 ##################################
60. **def** \_\_init\_\_(self,parent=None):
61. '''''构造函数'''
62. # 调用父类构造函数
63. super(MyButton,self).\_\_init\_\_(parent)
64. # 设置按钮尺寸
65. self.setFixedSize(QtCore.QSize(800,120))
66. # 设置按钮背景颜色
67. self.setStyleSheet('''''background-color:red;''')
68. ################################ 事件 #########################################
69. **def** mousePressEvent(self,event):
70. '''''鼠标按下事件'''
71. # 判断是否为鼠标左键按下
72. **if** event.button() == QtCore.Qt.LeftButton:
73. # 发射点击信号
74. self.clicked.emit(True)
75. # 传递至父窗口响应鼠标按下事件
76. self.parent().mousePressEvent(event)
78. ################################## 主函数 ##########################################
79. **if** \_\_name\_\_ == "\_\_main\_\_":
80. '''''主函数'''
81. # 声明变量
82. app = QtGui.QApplication(sys.argv)
83. # 创建窗口
84. window = MyWindow()
85. # 设置窗口显示
86. window.show()
87. #应用程序事件循环
88. sys.exit(app.exec\_())



看到这个截图，可以发现我们已经成功实现了点击按钮实现反编译的功能了。

### 0x6:总结

上面我们已经可以简单实现按钮的反编译了，毕竟没有白白浪费时间，证明我们是可以做一个反编译工具的，那么平时操作中，我们肯定不会是在代码中写死apk的路径的。常用的方法是加一个选择apk路径的window，然后选定apk，这样来确定apk所在的路径的。那么下一篇我们就一起来探索如何实现这个功能。

中秋第二天，下午就在探索中度过了。。。

祝福大家中秋愉快~~~

上面代码都是完整代码，所以本篇不再上传到网盘了，大家直接复制代码测试使用即可，欢迎有兴趣的大神交流指导~~~

破解群：，爱鬼哥

爱破解，爱鬼哥 ：377724636

# 跟着鬼哥学PyQtPart.3

<http://blog.csdn.net/guiguzi1110/article/details/39120939>

本文我们主要是创建一个window选择文件，从而来确定apk的路径，然后点击反编译按钮来对其进行反编译。

### 0x1:寻找相关api代码

QFileDialog类提供了类似的其他静态函数，如下表，通过这些函数，用户可以方便地定制自己的文件对话框。

静态函数 说明

getOpenFileName 获得用户选择的文件名

getSaveFileName 获得用户保存的文件名

getExistingDirectory 获得用户选择的已存在的目录名

getOpenFileNames 获得用户选择的文件名列表

### 0x2:按钮事件的总结

按钮的点击事件，经过一下午的分析，果断抛弃上一课的总结使用的鼠标点击事件，而直接对按钮的响应事件作出处理。也可以说通过今天下午的学习，比上午进步了一点点。。。

self.connect(decode,SIGNAL('clicked()'),self.apkD)

然后就是下午对布局文件的继续熟悉，感觉写布局代码越来越像写**[安卓](http://lib.csdn.net/base/android" \o "Android知识库" \t "_blank)**代码的布局了，好了，下面我贴上全部代码，里面已经写了比较详细的注释了，相信有编码基础的读者是可以看懂的：

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39120939) [copy](http://blog.csdn.net/guiguzi1110/article/details/39120939)

1. # -\*- coding: utf-8 -\*-
2. **from** PyQt4.QtGui **import** \*
3. **from** PyQt4.QtCore **import** \*
4. **import** sys
5. **import** os

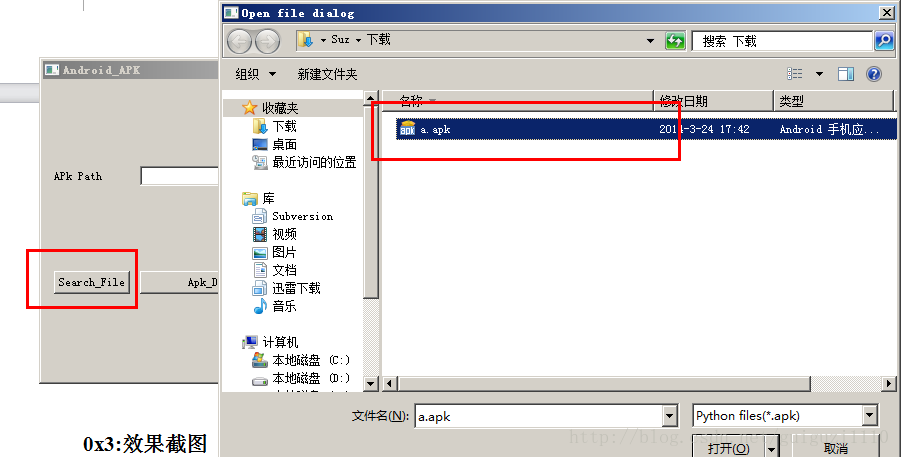
8. **class** GridLayout2(QWidget):
10. **def** \_\_init\_\_(self, parent=None):
11. QWidget.\_\_init\_\_(self, parent)
12. self.setWindowTitle('Android\_APK')
13. ###第一行
14. self.apk\_path = QLabel('APk Path')
15. self.apk\_path\_edit = QLineEdit()
17. ###第二行，三个按钮，这里第三个按钮的点击事件还没写
18. file\_search=QPushButton("Search\_File")
19. decode = QPushButton("Apk\_Decode")
20. build = QPushButton("Apk\_Build")
22. ###寻找文件按钮的点击事件
23. self.connect(file\_search,SIGNAL('clicked()'),self.openFile)
25. ###反编译按钮的点击事件
26. self.connect(decode,SIGNAL('clicked()'),self.apkD)
28. grid = QGridLayout()
29. grid.setSpacing(10)
30. grid.addWidget(self.apk\_path, 1, 0)
31. grid.addWidget(self.apk\_path\_edit, 1, 1)
33. grid.addWidget(file\_search,2,0)
34. grid.addWidget(decode, 2,1)
36. grid.addWidget(build, 2,2)
37. self.setLayout(grid)
38. self.resize(350, 300)

41. **def** openFile(self):
42. ##调用QfileDialog的相关方法来打开文件选择框
43. s=QFileDialog.getOpenFileName(self,"Open file dialog","/","Apk files(\*.apk)")
44. ##定义全局变量，用于接收需要反编译的apk的路径
45. **global** APK\_STR
46. APK\_STR=str(s)
47. self.apk\_path\_edit.setText(APK\_STR)
49. **def** apkD(self):
51. ###上面openFIle方法中定义的全局变量,打印一下路径，看看是否正确#####
52. **print**(APK\_STR)
54. ###apktool的命令使用#####
55. apkToD='java -jar apktool.jar d '+APK\_STR
57. ###调用os的system命令来执行cmd的相关命令###
58. os.system(apkToD)

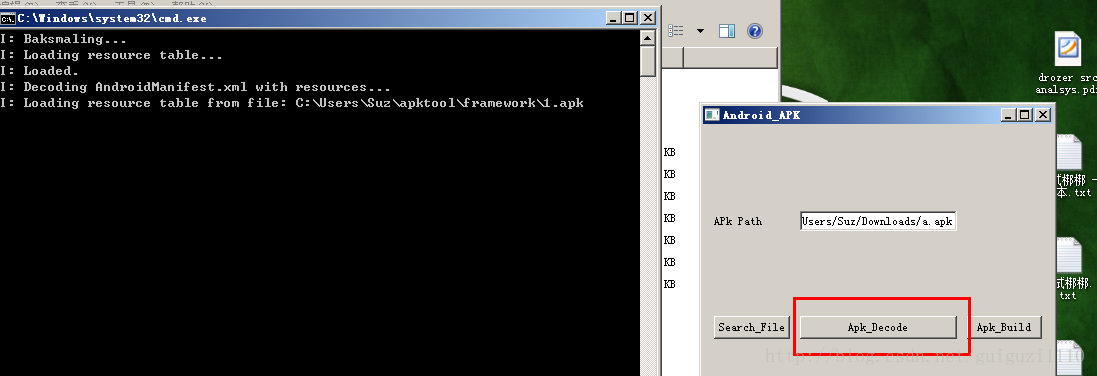

62. app = QApplication(sys.argv)
63. qb = GridLayout2()
64. qb.show()
65. sys.exit(app.exec\_())

### 0x3:效果截图

Search\_file的点击事件：



Apk\_Decode的点击事件：



### 0x4:总结

    上面的大家可以看到，我们已经可以实现两个功能了，即打开window找文件，然后进行反编译，但是在反编译完成以后，我们需要对里面的smali文件进行操作，然后再回编译文件，签名，才能对手机进行安装，所以下一篇文章，我们就主要是编写回编译的方法以及对apk签名的方法。

# 跟着鬼哥学PyQtPart.4

<http://blog.csdn.net/guiguzi1110/article/details/39136295>

经过前面三篇文章的学习，我们已经可以做出来一个简单的反编译apk的工具了。

为了更方便的使用我们自己编写的工具，所以我们对于第三篇文章中的search\_file button来进行处理，我们使用拖拽文件到QLineEdit控件上面来显示文件路径，而不用打开window来选择了，这样对我们的操作来说更加方便一点。

### 0x1:拖拽事件的处理

拖拽事件，经过一晚上的查询以及翻看[**Java**](http://lib.csdn.net/base/javase)已经写好的工具，终于找到dragEnterEvent和dropEvent两个事件。关于这两个事件的详细介绍，自行google搜索一下即可。

要想使用这个事件，我们需要重写一下QLineEdit控件，并且在其init方法里面设置setAcceptDrops为True来接收拖拽事件。

所以现在的问题就是我们如何来获取得到拖入文件的路径呢？

经过查询mimeData有一些方法可以参考，经过**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**，urls()属性还是比较贴近我们所需的，但是还需要对其进行处理得到文件路径。

详细代码如下：

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39136295) [copy](http://blog.csdn.net/guiguzi1110/article/details/39136295)

1. **def** dropEvent(self, event):
3. ###获取拖放过来的文件的路径
5. st=str(event.mimeData().urls())
7. **print**(st)
9. ###这里没发现好的api，所以用mimeData()的urls属性了，text()不好用，所以下面就是字符串截取，来还原路径
11. a='///'
13. b=')]'

16. num2=st.index(b)-1
18. num=st.index(a)+3
20. st=st[num:num2]
22. ##这里定义全局变量，拖入文件或者文件夹的路径
24. **global** APK\_STR
26. APK\_STR=st
28. self.setText(st)

### 0x2:重新做一下布局文件

详细代码如下，有详细注释，这里不再多说：

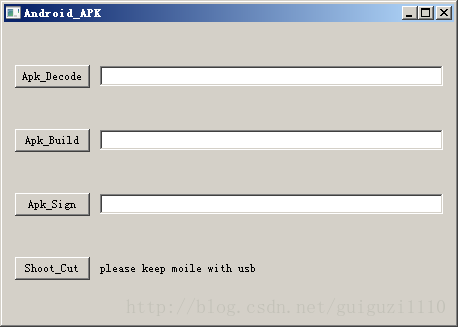
**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39136295) [copy](http://blog.csdn.net/guiguzi1110/article/details/39136295)

1. QWidget.\_\_init\_\_(self, parent)
2. self.setWindowTitle('Android\_APK')
3. ###第一行
4. self.apk\_decode=QPushButton("Apk\_Decode")
5. self.apk\_path\_edit = MyEditText(self)
7. ###第二行，三个按钮，这里第三个按钮的点击事件还没写
8. self.apk\_build=QPushButton("Apk\_Build")
9. self.apk\_build\_edit = MyEditText()
11. ###第三行
12. self.sign = QPushButton("Apk\_Sign")
13. self.apk\_sign\_edit = MyEditText()
15. ##第四行
16. self.shootcut=QPushButton("Shoot\_Cut")
17. self.shootcut\_text=QLabel("please keep moile with usb")

20. ###布局文件的编写，四行两列
21. grid = QGridLayout()
22. grid.setSpacing(10)
24. grid.addWidget(self.apk\_decode, 1, 0)
25. grid.addWidget(self.apk\_path\_edit, 1, 1)
27. grid.addWidget(self.apk\_build, 2, 0)
28. grid.addWidget(self.apk\_build\_edit, 2, 1)
30. grid.addWidget(self.sign,3,0)
31. grid.addWidget(self.apk\_sign\_edit, 3,1)
33. grid.addWidget(self.shootcut,4,0)
34. grid.addWidget(self.shootcut\_text, 4,1)

37. ###设置窗口的高宽，将gridview放入大布局中
38. self.setLayout(grid)
39. self.resize(450, 300)

效果图如下：



### 0x3:整合相关按钮的点击事件

前面一篇文章已经将反编译的按钮事件做好，所以这里我们照着写一下回编译和签名以及最后的截屏事件即可。

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39136295) [copy](http://blog.csdn.net/guiguzi1110/article/details/39136295)

1. **def** apkD(self):
3. ###上面openFIle方法中定义的全局变量,打印一下路径，看看是否正确#####
4. **print**(APK\_STR)
6. ###apktool的命令使用#####
7. apkToD='java -jar apktool.jar d '+APK\_STR
9. ###调用os的system命令来执行cmd的相关命令###
10. os.system(apkToD)
12. **def** apkB(self):
14. ###上面openFIle方法中定义的全局变量,打印一下路径，看看是否正确#####
15. **print**(APK\_STR)
17. ###apktool的命令使用#####
18. apkToD='java -jar apktool.jar b '+APK\_STR
20. ###调用os的system命令来执行cmd的相关命令###
21. os.system(apkToD)
23. **def** apkS(self):
25. apkSign='java -jar signapk.jar testkey.x509.pem testkey.pk8 '+APK\_STR+' Signed\_apk.apk'
27. **print** apkSign
29. ###调用os的system命令来执行cmd的相关命令###
30. os.system(apkSign)

33. **def** shootCut(self):
35. pre\_com='adb wait-for-device'
36. os.system(pre\_com)
38. ###将截图保存到local下
39. fir\_com='adb shell screencap -p /data/local/tmp/tmp.png'
41. **print** (fir\_com)
43. os.system(fir\_com)
45. ###当前时间，精确到秒，来保存图片命名
46. timestamp = time.strftime('%Y-%m-%d-%H-%M-%S',time.localtime(time.time()))
48. path=os.getcwd()
50. ###将其导出来
51. sec\_com='adb pull /data/local/tmp/tmp.png '+path+'\\'+timestamp+'.png'
53. **print**(sec\_com)
54. os.system(sec\_com)
56. thi\_com='adb shell rm /data/local/tmp/tmp.png'
58. os.system(thi\_com)

### 0x4:完整的代码

下面附上完整的代码，有兴趣的朋友，直接测试即可。记得先配置好pyqt相关开发环境。

**[python]** [view plain](http://blog.csdn.net/guiguzi1110/article/details/39136295) [copy](http://blog.csdn.net/guiguzi1110/article/details/39136295)

1. # -\*- coding: utf-8 -\*-
2. **from** PyQt4.QtGui **import** \*
3. **from** PyQt4.QtCore **import** \*
4. **import** sys
5. **import** os
6. **import** time

9. **class** MainLayout(QWidget):
11. **def** \_\_init\_\_(self, parent=None):
13. QWidget.\_\_init\_\_(self, parent)
14. self.setWindowTitle('Android\_APK')
15. ###第一行
16. self.apk\_decode=QPushButton("Apk\_Decode")
17. self.apk\_path\_edit = MyEditText(self)
19. ###第二行，三个按钮，这里第三个按钮的点击事件还没写
20. self.apk\_build=QPushButton("Apk\_Build")
21. self.apk\_build\_edit = MyEditText()
23. ###第三行
24. self.sign = QPushButton("Apk\_Sign")
25. self.apk\_sign\_edit = MyEditText()
27. ##第四行
28. self.shootcut=QPushButton("Shoot\_Cut")
29. self.shootcut\_text=QLabel("please keep moile with usb")
31. ###反编译按钮的点击事件
32. self.connect(self.apk\_decode,SIGNAL('clicked()'),self.apkD)
34. ###回编译按钮的点击事件
35. self.connect(self.apk\_build,SIGNAL('clicked()'),self.apkB)
37. ###签名按钮的点击事件
38. self.connect(self.sign,SIGNAL('clicked()'),self.apkS)
40. ###截屏按钮的点击事件
41. self.connect(self.shootcut,SIGNAL('clicked()'),self.shootCut)

44. ###布局文件的编写，四行两列
45. grid = QGridLayout()
46. grid.setSpacing(10)
48. grid.addWidget(self.apk\_decode, 1, 0)
49. grid.addWidget(self.apk\_path\_edit, 1, 1)
51. grid.addWidget(self.apk\_build, 2, 0)
52. grid.addWidget(self.apk\_build\_edit, 2, 1)
54. grid.addWidget(self.sign,3,0)
55. grid.addWidget(self.apk\_sign\_edit, 3,1)
57. grid.addWidget(self.shootcut,4,0)
58. grid.addWidget(self.shootcut\_text, 4,1)

61. ###设置窗口的高宽，将gridview放入大布局中
62. self.setLayout(grid)
63. self.resize(450, 300)

66. ''''''def openFile(self):
67. ##调用QfileDialog的相关方法来打开文件选择框
68. s=QFileDialog.getOpenFileName(self,"Open file dialog","/","Apk files(\*.apk)")
69. ##定义全局变量，用于接收需要反编译的apk的路径
70. global APK\_STR
71. APK\_STR=str(s)
72. self.apk\_path\_edit.setText(APK\_STR)'''
74. **def** apkD(self):
76. ###上面openFIle方法中定义的全局变量,打印一下路径，看看是否正确#####
77. **print**(APK\_STR)
79. ###apktool的命令使用#####
80. apkToD='java -jar apktool.jar d '+APK\_STR
82. ###调用os的system命令来执行cmd的相关命令###
83. os.system(apkToD)
85. **def** apkB(self):
87. ###上面openFIle方法中定义的全局变量,打印一下路径，看看是否正确#####
88. **print**(APK\_STR)
90. ###apktool的命令使用#####
91. apkToD='java -jar apktool.jar b '+APK\_STR
93. ###调用os的system命令来执行cmd的相关命令###
94. os.system(apkToD)
96. **def** apkS(self):
98. apkSign='java -jar signapk.jar testkey.x509.pem testkey.pk8 '+APK\_STR+' Signed\_apk.apk'
100. **print** apkSign
102. ###调用os的system命令来执行cmd的相关命令###
103. os.system(apkSign)

106. **def** shootCut(self):
108. pre\_com='adb wait-for-device'
109. os.system(pre\_com)
111. ###将截图保存到local下
112. fir\_com='adb shell screencap -p /data/local/tmp/tmp.png'
114. **print** (fir\_com)
116. os.system(fir\_com)
118. ###当前时间，精确到秒，来保存图片命名
119. timestamp = time.strftime('%Y-%m-%d-%H-%M-%S',time.localtime(time.time()))
121. path=os.getcwd()
123. ###将其导出来
124. sec\_com='adb pull /data/local/tmp/tmp.png '+path+'\\'+timestamp+'.png'
126. **print**(sec\_com)
127. os.system(sec\_com)
129. thi\_com='adb shell rm /data/local/tmp/tmp.png'
131. os.system(thi\_com)
133. **class** MyEditText(QLineEdit):
135. **def** \_\_init\_\_(self,parent=None):
137. super(MyEditText,self).\_\_init\_\_(parent)
139. #这里在init初始化，接受拖入事件
140. self.setAcceptDrops(True)

143. ###重写两个拖入文件的操作方法
144. **def** dragEnterEvent(self,event):
146. event.accept()

149. **def** dropEvent(self, event):
151. ###获取拖放过来的文件的路径
153. st=str(event.mimeData().urls())
155. **print**(st)
157. ###这里没发现好的api，所以用mimeData()的urls属性了，text()不好用，所以下面就是字符串截取，来还原路径
159. a='///'
161. b=')]'

164. num2=st.index(b)-1
166. num=st.index(a)+3
168. st=st[num:num2]
170. ##这里定义全局变量，拖入文件或者文件夹的路径
172. **global** APK\_STR
174. APK\_STR=st
176. self.setText(st)

179. ###程序的主入口
180. **if** \_\_name\_\_ == "\_\_main\_\_":
181. app = QApplication(sys.argv)
182. qb = MainLayout()
183. qb.show()
184. sys.exit(app.exec\_())

效果图，前面几篇基本都有了，这里不再贴出了。

### 注意：

1.反编译采用的是apktool.jar的命令，所以得将其放到.py文件对应目录

2.签名采用google的公钥来过的，所以其文件也必须在同目录下

3.截屏动作，得保证手机正常连接usb，adb进程不被占用，所以如果读者有自己编译的adb是最好的，若此功能不好用，直接做个cmd脚本将上述方法中的命令放入即可

4.这四篇文章，仅记录一下PyQt的相关学习过程，读者可以随意对其进行扩展开发，做一个IDE工具集合搜索，修改，保存以及一些自动化工具

### 结束语：

虽然中秋几天假期没有出去玩，不过从头开始学习PyQt还是有一些进步的。代码方面编写肯定是有很多疏漏，其对应功能方面也可以做更多优化，有兴趣的朋友，请继续加油。

仅以一句话勉励自己：

勤能补拙是良训，一分辛苦一分才。

相关附件之类的，我就上传到百度网盘了：

<http://pan.baidu.com/s/1o6sONoQ>

