QT图形化软件设计

0、教程

1. 入门视频教程： 了解大致开发流程、QT designer的使用
2. Qt for Python 文档 ： QT介绍、案例、C-python接口等
3. QT for C/C++ 文档
4. QT官网

1、安装相应的库

现在python开发QT界面，有PyQT5和Pyside2，前者是第三方开发的库，由于出现的时间较早，所以也比较流行,由于后者是QT官方库，所有下面的学习都基于Pyside2。

```
pip install pyside2
```

2、运行Qt Designer

Qt Designer 是Pyside2自带的一个可视化UI设计软件，实现和VB一样的界面开发，即所见即所得。

1. **打开Anaconda Prompt ，进入装有Pyside2的虚拟环境，运行Qt Designer**

```
designer.exe
```

保存的UI界面文件以 .ui 后缀保存，其内容是XML格式的。

2.1、加载ui文件

```
# File: main.py
import sys
from PySide2.QtUiTools import QUiLoader
from PySide2.QtWidgets import QApplication
from PySide2.QtCore import QFile, QIODevice
if __name__ == "__main__":
 app = QApplication(sys.argv)
 ui_file_name = "mainwindow.ui"
 ui_file = QFile(ui_file_name)
 if not ui_file.open(QIODevice.ReadOnly):
     print("Cannot open {}: {}".format(ui_file_name,
ui_file.errorString()))
     sys.exit(-1)
 loader = QUiLoader()
 window = loader.load(ui_file)
 ui_file.close()
 if not window:
     print(loader.errorString())
     sys.exit(-1)
 window.show()
 sys.exit(app.exec_())
```

2.2、.ui 转化为 .py

- 如果需要的话，.ui 文件可以转化为 .py 文件，即python语言定义的界面。
- 一般在最终版本的时候可以转化为py文件，这样打包成可执行文件为更方便。

生成python class，直接在终端运行：

```
pyside2-uic mainwindow.ui > ui_mainwindow.py
```

使用：

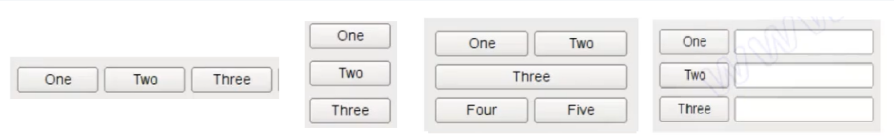
```
import sys
from PySide2.QtWidgets import QApplication, QMainWindow
from PySide2.QtCore import QFile
from ui_mainwindow import Ui_MainWindow  # 从转化后的python文件导
入UI

```

```
6
7       class MainWindow(QMainWindow):
8           def __init__(self):
9               super(MainWindow, self).__init__()
10              self.ui = Ui_MainWindow()
11              self.ui.setupUi(self)
12
13
14      if __name__ == "__main__":
15          # app = QApplication(sys.argv)
16          app = QApplication()
17          window = MainWindow()
18          window.show()
19          sys.exit(app.exec_())
```

3、常用控件

> QT官网控件 介绍

3.1、 Layout

> 我们如果希望自己的控件能跟随窗口自动改变大小，就需要将控件放到一个 layout 中。建议先看 视频 的操作，否则自己模式会很麻烦，因为会会忽略很多非常细但好用的操作。
>
> 常用的 layout 类型：
>
> - QHBoxLayout : 水平布局
> - QVBoxLayout : 垂直布局
> - QGridLayout : 表格布局
> - QFormlayout : 表单布局
>
> 
>
>
> 布局之间可以嵌套！因此如果嵌套顺序不对，操作起来非常麻烦， 布局操作步骤经验：
>
> 1. 先不用任何布局layout， 先把所有控件按位置摆放到界面上。
> 2. 然后从内到外开始，进行控件的Layout设置，从而实现布局嵌套

3. 最后调整Layout中控件的大小比例，优先使用Layout的 layoutStrentch 属性来控制，可用Spaces来补充不显示的控件。

3.2、绘图相关

1. QPainter

   QPainter提供了高度优化的功能来完成GUI程序所需的大部分绘图。它可以画任何东西，从简单的线条到复杂的形状，如馅饼和和弦。它还可以绘制对齐文本和像素图。通常，它在一个自然的坐标系中绘制，但是它也可以进行视图和世界的转换。QPainter可以操作任何对象,继承了QPaintDevice类

2. Qt 为处理图像数据提供了 4 个类： `QImage` ， `QPixmap` ， `QBitmap` and `QPicture` .

3. QGraphicsView

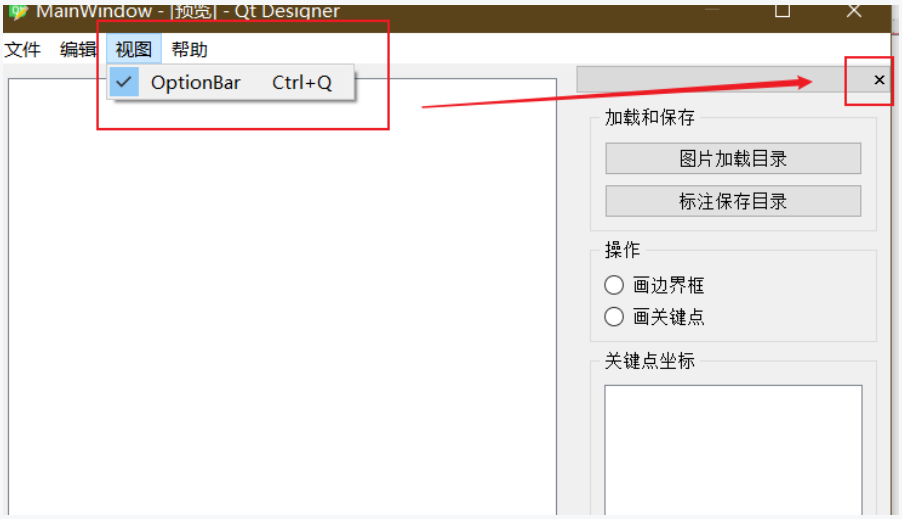
   

4、 Signal & slot

:使用例子

在QT Designal中也可以设置信号和槽，当我们对信号发送者进行某种操作时，会产生一种信号；信号接收者接收到信号，会执行相应的操作，这个操作就是槽。

如上，我添加了一个操作，就是视图中可选的OptionBar，其选中状态与dockWidget是否显示的状态关联起来。

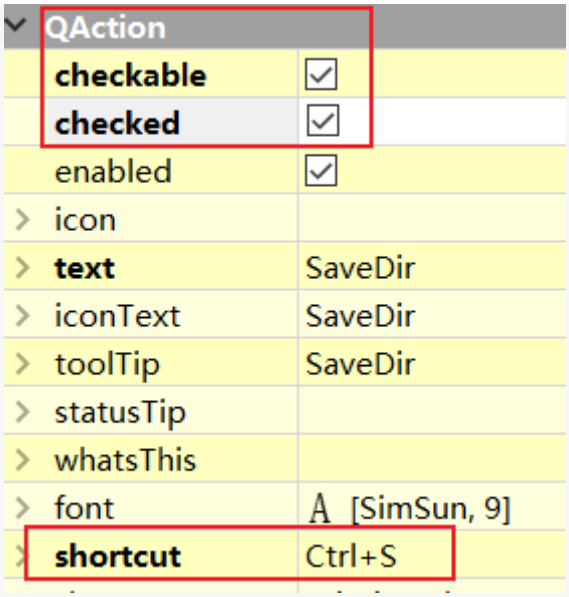


- 信号发送者是菜单中视图的可选项OptionBar的Action，其操作为toggled(bool)，toggled表示开关，bool表示每次操作，会改变该可选项的真/假，这里表示是否选中。
- 信号接收者是dockerWidget，槽是setVisible(bool)，表示收到信号后会改变显示状态的真/假。

其中，在菜单中添加了OptionBar后，怎么把它设置为可选的，并添加快捷键？

1. 在动作编辑器定义可选项OptionBar的Action。

2. 在属性中定义。



5、发布软件

写好程序后，生成可执行文件，就可以不用再安装运行环境的情况下，在其他电脑上运行。

这里用 pyinstalller 库 来打包生成可执行文件. 参考资料

1. 安装

```
pip install pyinstaller
```

2. 打包

```
pyinstaller -F main.py -w --add-data="your.ui;ui_path" --hidden-import PySide2.QtXml --icon="图标路径"
# 一般第一次打包时，不加 -w 参数，从而可以从终端查看报错信息
# --noconsole -w 不显示控制台，控制台一般是调试时显示
# --hidden-import 打包时，要把动态加载的库（如打开某些文件等），也一并打包（一般报错时才知道）。
# --icon -i 指定软件图标，必须为 ico文件
# 记得把.ui和.ico文件放到dist最终目录中，build目录只是临时目录而已。
pyinstaller -F main.py --add-data="your.ui;ui_path" --hidden-import PySide2.QtXml --icon="图标路径"
# example
pyinstaller -w -F main.py --add-data="./UI/labelKPs.ui;./UI" --hidden-import PySide2.QtXml --icon="./hand_icon.ico"
```

```
Beginning shutdown...
Traceback (most recent call last):
  File "main.py", line 6, in <module>
  File "C:\Users\WISE&B~1\AppData\Local\Temp\embedded.0r

  File "D:\python\anaconda3\envs\labelme\lib\site-packag
module
    exec(bytecode, module.__dict__)
  File "app.py", line 4, in <module>
  File "C:\Users\WISE&B~1\AppData\Local\Temp\embedded.0r

ImportError: could not import module 'PySide2.QtXml'
[2096] Failed to execute script main
```