

## PyQt 制作 GUI 实战：通过制作小而美的计算器学会使用 ...

前段时间，我们使用 html+css+javascript 前端技术，后端使用 Flask 设计过一款精美的 web 版计算器，它能实现计算优先级的自动提升。

今天与大家一起实现一款桌面版计算器，使用 pyqt 设计器，设计整个 app 界面。

### 第一步：熟悉设计器

使用 qt designer ，按装 anaconda 后，在如下路径找到 qt designer：

□

它使用方便，能实现控件拖拽，布局也更加灵活，远胜于 tkinter 控件库。如下是它的整个软件界面，左侧是常用的控件对象。

□

根据弹出的对话框，创建窗体，命名为 `XiaoDing`。

设计完成后的界面显示如下：

□

下面解释 `XiaoDing` 界面的设计过程。

qt designer 提供的常用控件基本都能满足开发需求，通过拖动左侧的控件，很便捷的就能搭建出如下的 UI 界面，比传统的手写控件代码要方便很多。

选择左上菜单，点击新建窗体，然后选择 `Main Window`

□

点击创建按钮，出现下面的 GUI 界面：

□

看到右上窗口的对象查看器，此时根对象为 `MainWindow`，子对象有 `centralwidget`，类型为 `QWidget`，`menubar` 和 `statusbar` 对象。

□

中间是属性设置窗口，上面提到 4 个对象都显示在这里，通过界面，我们可以很方便地修改对象的属性。

□

修改 Window 的 title 为 `XiaoDing`:

□

### 第二步：添加控件

左侧控件筛选框里输入 `QLCDN`，自动就筛选出这个控件，然后拖动到界面中。

□

手动调整控件大小：

□

下面再选择按钮控件 `QPushButton`，依次推动计算器界面的所有按钮控件，布局完成后的界面如下所示：

□

注意布局界面时，可以使用 `Grid Layout` 控件，能方便我们对齐各个按钮：

□

右上角的对象生成树结构为：

□

保存 ui 文件。

### 第三步：转换为 py 文件

转换 ui 布局文件为 py 文件。

使用如下命令，将设计好的 ui 文件转为 py 文件

```
pyuic5 -o ./calculator/MainWindow.py ./calculator/mainwindow.ui
```

复制

至此，完成 GUI 的设计部分。

### 第四步：编写槽函数

下面开始编写槽函数，实现计算器的业务逻辑。

首先看一下我们上步生成的 py 文件里的一个重要类 `Ui_MainWindow`，里面就是实现窗体布局的后台代码。

下面是选取前十几行代码，看到包括布局相关对象，控件相关对象等。

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(503, 433)
        self.centralWidget = QtWidgets.QWidget(MainWindow)
```

复制

#### 第一步：熟悉设计器

第二步：添加控件

第三步：转换为 py 文件

第四步：编写槽函数

小结

```
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,
QtWidgets.QSizePolicy.Maximum)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.centralWidget.sizePolicy()).hasHeightForWidth()
self.centralWidget.setSizePolicy(sizePolicy)
self.centralWidget.setObjectName("centralWidget")
self.verticalLayout = QtWidgets.QVBoxLayout(self.centralWidget)
self.verticalLayout.setContentsMargins(11, 11, 11, 11)
self.verticalLayout.setSpacing(6)
self.verticalLayout.setObjectName("verticalLayout")
self.lcdNumber = QtWidgets.QLCDNumber(self.centralWidget)
self.lcdNumber.setDigitCount(10)
self.lcdNumber.setObjectName("lcdNumber")
self.verticalLayout.addWidget(self.lcdNumber)
self.gridLayout = QtWidgets.QGridLayout()
```

下面开始编写计算器的业务逻辑代码，首先导入需要的库：

```
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

import operator

from MainWindow import Ui_MainWindow
```

两个全局变量，表示当前计算器的输入状态值。

```
# 计算器的状态值
READY = 0
INPUT = 1
```

定义 `MainWindow` 类，继承自两个类：`QMainWindow`，和我们界面设计完成后刚才看到的类 `Ui_MainWindow`

```
class MainWindow(QMainWindow, Ui_MainWindow):
    def __init__(self, *args, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
```

为数字按键安装槽函数

```
# 为数字按键安装槽函数
for n in range(0, 10):
    getattr(self, 'pushButton_n%s' % n).pressed.connect(lambda v=
n: self.input_number(v))
```

为按钮安装槽函数

```
# 为按钮安装槽函数
self.pushButton_add.pressed.connect(lambda: self.operation(operat
or.add))
self.pushButton_sub.pressed.connect(lambda: self.operation(operat
or.sub))
self.pushButton_mul.pressed.connect(lambda: self.operation(operat
or.mul))
self.pushButton_div.pressed.connect(lambda: self.operation(operat
or.truediv))
self.pushButton_pc.pressed.connect(self.operation_pc)
self.pushButton_eq.pressed.connect(self.equals)
self.actionReset.triggered.connect(self.reset)
self.pushButton_ac.pressed.connect(self.reset)
self.actionExit.triggered.connect(self.close)
self.pushButton_m.pressed.connect(self.memory_store)
self.pushButton_mr.pressed.connect(self.memory_recall)
```

初始化操作还包括，初始化计算器的所有状态变量等。

```
self.memory = 0
self.reset()
self.show()
```

重置到计算器初始状态的行为如下：

```
def reset(self):
    self.state = READY
    self.stack = [0]
    self.last_operation = None
    self.current_op = None
    self.display()
```

`display` 在计算器的数字显示窗口，显示当前 `stack` 列表的最后一个数字：

```
def display(self):
    self.lcdNumber.display(self.stack[-1])
```

`stack` 列表内初始化只有一个数字，为 0。

`memory_store` 方法内存中记忆着数字显示屏上的值

```
def memory_store(self):
    self.memory = self.lcdNumber.value()
```

`memory_recall` 方法会将记忆值推到 `stack` 列表中，同时显示在数字显示屏中。

```
def memory_recall(self):
    self.state = INPUT
    self.stack[-1] = self.memory
    self.display()
```

`input_number` 方法实现数字显示屏界面数字的持续输入，注意第一次输入时，此时 `state` 值为 `READY`，直接将用户输入的 `v` 值推到 `stack` 列表，同时修改 `state` 为 `INPUT` 状态，当用户继续输入第 2 个数字时，此时 `state` 变为 `INPUT`，`self.stack[-1] * 10 + v` 实现将第 2 位数字推入到 `stack` 列表中，依次类推。

```
def input_number(self, v):
    if self.state == READY:
        self.state = INPUT
        self.stack[-1] = v
    else:
        self.stack[-1] = self.stack[-1] * 10 + v

    self.display()
```

复制

operation 方法实现加减乘除操作，推入到 stack 列表中。

```
def operation(self, op):
    if self.current_op:
        self.equals()

    self.stack.append(0)
    self.state = INPUT
    self.current_op = op
```

复制

operation\_pc 实现当前 stack 列表的最后一个元除以 100 的功能。

```
def operation_pc(self):
    self.state = INPUT
    self.stack[-1] *= 0.01
    self.display()
```

复制

equals 方法计算当前结果

```
def equals(self):
    if self.state == READY and self.last_operation:
        s, self.current_op = self.last_operation
        self.stack.append(s)

    if self.current_op:
        self.last_operation = self.stack[-1], self.current_op

    try:
        self.stack = [self.current_op*(self.stack)]
    except Exception:
        self.lcdNumber.display('Errr')
        self.stack = [0]
    else:
        self.current_op = None
        self.state = READY
        self.display()
```

复制

计算器的 Main 函数：

```
if __name__ == '__main__':
    app = QApplication([])
    app.setApplicationName("XiaoDing")

    window = MainWindow()
    app.exec_()
```

复制

最后计算器的启动界面，如下图所示，大家动手实践一遍吧。

□

小结

今天通过qt designer 设计一个小而美的计算器，然后编写槽函数，实现计算器的业务逻辑，希望通过这个计算器实战项目，大家能入门 pyqt5 的 GUI 开发。

附项目的完整代码下载链接如下：

链接: <https://pan.baidu.com/s/1M5K3GDIL3z73vzohi4aYKA> 提取码: 4gde

互动评论



说点什么

评论



Ryoko

2 个月前

pressed和clicked有何区别呢

鼓掌



存

1

<