# 1.安装PyQt5

## 1.网上说是这么安装的

|  |
| --- |
| pip install PyQt5  pip install pyqt5-tools |

但是我在安装pyqt5-tools时发现一个问题，就是pyqt5-tools安装的时候会先安装PyQt5，如果他发现已经安装了pyqt5，他会把pyqt5卸载了然后重新安装，

## 2.所以其实只需要这么干

|  |
| --- |
| pip install pyqt5-tools |

## 就会安装所有的组件

## 3.pyqt5安装好后，designer.exe是在：%PYTHON\_HOME%\Lib\site-packages\qt5\_applications\Qt\bin里面的

|  |
| --- |
| 如果你安装的是python3.8，它就在：D:\programs\Python38\Lib\site-packages\qt5\_applications\Qt\bin如果你安装的是python3.10，它就在：D:\programs\Python310\Lib\site-packages\qt5\_applications\Qt\bin |

# 2. PyCharm配置环境

启动PyCharm后，新建一个PyQt5空项目后，需要配置Qt Designer、pyuic、pyrcc工具，相关配置方法如下：

## 1）配置 Qt Designer

Qt Designer 是通过拖拽的方式放置控件，并实时查看控件效果进行快速UI设计。

最终生成.ui文件（实质上是XML格式的文件），可以通过pyuic5工具转换成.py文件。

在Pycharm中，依次打开 File - Settings - Tools - External Tools，点击 + Create Tool，配置如下：

|  |
| --- |
| Name: QtDisigner  Program : D:\programs\Python310\Lib\site-packages\qt5\_applications\Qt\bin\designer.exe # 当前designer目录，请根据实际修改  Working directory: $ProjectFileDir$ ，注意不要写$FileDir$,会说找不到目录启动失败 |
|  |

## 2）在PyCharm主界面，依次点击 Tools - External Tools - QtDisigner，即可启动Qt Disigner

|  |
| --- |
|  |

## 3）PyUIC配置

PyUIC主要是把Qt Designer生成的.ui文件换成.py文件。Name: PyUIC

在Pycharm中，依次打开 File - Settings - Tools - External Tools，点击 + Create Tool，配置如下：

|  |
| --- |
| Program : D:\programs\Python310\Scripts\pyuic5.exe  Arguments: $FileName$ -o $FileNameWithoutExtension$.py  Working directory: $ProjectFileDir$ |
|  |

## 4） PyRCC配置

PyRCC主要是把编写的.qrc资源文件换成.py文件。

在Pycharm中，依次打开 File - Settings - Tools - External Tools，点击 + Create Tool，配置如下：

|  |
| --- |
| Name: PyRCC  Program : D:\programs\Python310\Scripts\pyrcc5.exe # 当前rcc工具目录，请根据实际修改  Arguments: $FileName$ -o $FileNameWithoutExtension$\_rc.py  Working directory: （$FileDir$）$ProjectFileDir$ |
|  |

# 3.使用

1. 下面我们以一个简单的城市天气预报为例，演示使用PyQt5开发一个GUI程序的基本流程。

## 3.1 获取天气数据

主要逻辑是通过Http接口调用免费的API接口获取相关城市天气数据，详见[天气API说明](https://www.sojson.com/api/weather.html)

如测试一下请求天津的天气，链接为：<http://t.weather.sojson.com/api/weather/city/101030100>

返回成功状态（status）为：200 ，失败为非200，返回数据为json数据，直接解析获取即可。

### 1） 设计界面UI

打开Qt Designer，可参考下图设计Weather.ui:

|  |
| --- |
|  |

### 2）我们主要用到的控件有Button, GroupBox, Label,ComboBox,TextEdit，同时定义了两个按钮queryBtn及clearBtn，分别用来查询及清空天气数据。我们需要绑定槽函数，方法如下：

在Qt Designer右下角选择 信号/槽编辑器，点击+号新增

分别选择queryBtn及clearBtn，选择信号 clicked(), 接收者 Dialog 及槽 accept() （我没找到绑定自定义槽函数的方法...）

|  |
| --- |
|  |

### 3）选择保存为 Weather.ui文件

## 3.2转换.ui文件为.py文件

在PyCharm中选中Weather.ui文件后，右键选择 External Tools - PyUIC，即可生成Weather.py，

其中，我们需要把两个按钮绑定的槽函数：

|  |
| --- |
| self.queryBtn.clicked.connect(Dialog.accept)  self.clearBtn.clicked.connect(Dialog.accept) |

修改为自定义函数：

|  |
| --- |
| self.queryBtn.clicked.connect(Dialog.queryWeather)  self.clearBtn.clicked.connect(Dialog.clearText) |

### 最终Weather.py内容如下：

|  |
| --- |
| *# -\*- coding: utf-8 -\*-  # Form implementation generated from reading ui file 'Weather.ui' # # Created by: PyQt5 UI code generator 5.15.9 # # WARNING: Any manual changes made to this file will be lost when pyuic5 is # run again. Do not edit this file unless you know what you are doing.* **from** PyQt5 **import** QtCore, QtGui, QtWidgets   **class** Ui\_Dialog(object):  **def** setupUi(self, Dialog):  Dialog.setObjectName(**"Dialog"**)  Dialog.resize(581, 369)  self.groupBox = QtWidgets.QGroupBox(Dialog)  self.groupBox.setGeometry(QtCore.QRect(10, 10, 561, 271))  self.groupBox.setObjectName(**"groupBox"**)  self.label = QtWidgets.QLabel(self.groupBox)  self.label.setGeometry(QtCore.QRect(30, 30, 54, 12))  self.label.setObjectName(**"label"**)  self.comboBox = QtWidgets.QComboBox(self.groupBox)  self.comboBox.setGeometry(QtCore.QRect(90, 30, 69, 22))  self.comboBox.setObjectName(**"comboBox"**)  self.comboBox.addItem(**""**)  self.comboBox.addItem(**""**)  self.comboBox.addItem(**""**)  self.textEdit = QtWidgets.QTextEdit(self.groupBox)  self.textEdit.setGeometry(QtCore.QRect(30, 60, 521, 191))  self.textEdit.setObjectName(**"textEdit"**)  self.btnQuery = QtWidgets.QPushButton(Dialog)  self.btnQuery.setGeometry(QtCore.QRect(90, 310, 75, 23))  self.btnQuery.setObjectName(**"btnQuery"**)  self.btnClear = QtWidgets.QPushButton(Dialog)  self.btnClear.setGeometry(QtCore.QRect(340, 310, 75, 23))  self.btnClear.setObjectName(**"btnClear"**)   self.retranslateUi(Dialog)  self.btnQuery.clicked.connect(Dialog.queryWeather) *# type: ignore* self.btnClear.clicked.connect(Dialog.clearText) *# type: ignore* QtCore.QMetaObject.connectSlotsByName(Dialog)   **def** retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate(**"Dialog"**, **"Dialog"**))  self.groupBox.setTitle(\_translate(**"Dialog"**, **"城市天气预报"**))  self.label.setText(\_translate(**"Dialog"**, **"城市"**))  self.comboBox.setItemText(0, \_translate(**"Dialog"**, **"北京"**))  self.comboBox.setItemText(1, \_translate(**"Dialog"**, **"上海"**))  self.comboBox.setItemText(2, \_translate(**"Dialog"**, **"天津"**))  self.btnQuery.setText(\_translate(**"Dialog"**, **"查询"**))  self.btnClear.setText(\_translate(**"Dialog"**, **"清空"**)) |

## 3.3 调用MainDialog

在MainDialog中调用界面类Ui\_Dialog，然后在其中中添加查询天气的业务逻辑代码，这样就做到了界面显示和业务逻辑的分离。

新增pyqt\_get\_weather.py文件， 中文程序的主文件，在这个文件里面加载Weather.py界面文件。在MainDialog类中定义了两个槽函数queryWeather()和clearText(),以便在界面文件Weather.ui中定义的两个按钮(queryBtn 和clearBtn) 触发clicked 信号与这两个槽函数进行绑定。

### 完整代码如下:

|  |
| --- |
| *# -\*- encoding:utf-8 -\*- # @日期和时间：2024/3/20 17:02 # @Author: Kenny cai # @File name:pyqt\_get\_weather.py # @dev tool:PyCharm # pyqt查询天气的小demo,需要使用request库* **import** sys **import** Weather **from** PyQt5.QtWidgets **import** QApplication,QDialog **import** requests  **class** MainDialog(QDialog):  **def** \_\_init\_\_(self,parent=**None**):  super(MainDialog, self).\_\_init\_\_(parent)  self.ui = Weather.Ui\_Dialog()  self.ui.setupUi(self)   **def** queryWeather(self):  cityName = self.ui.comboBox.currentIndex() *# 从combobox中获取需要查询天气的城市* cityCode = self.getCode(cityName)  r = requests.get(**"http://t.weather.sojson.com/api/weather/city/{}"**.format(cityCode))   print(r.json())   *#查询到数据就显示在文本框里面* ret = r.json()  **if** ret.get(**'status'**) == 200:  weatherMsg = **f"城市：{**ret[**'cityInfo'**][**'city'**]**}\n日期：{**ret[**'data'**][**'forecast'**][0][**'ymd'**]**}\n天气：{**ret[**'data'**][**'forecast'**][0][**'type'**]**}\nPM 2.5：{**int(ret[**'data'**][**'pm25'**])**} {**ret[**'data'**][**'quality'**]**}\n温度：{**ret[**'data'**][**'wendu'**]**}\n湿度：{**ret[**'data'**][**'shidu'**]**}\n风力：{**ret[**'data'**][**'forecast'**][0][**'fl'**]**}\n\n{**ret[**'data'**][**'forecast'**][0][**'notice'**]**}"** *#查询不到显示“查询失败稍后在试”* **else**:  weatherMsg =**"查询失败，稍后再试"** self.ui.textEdit.setText(weatherMsg)   **def** getCode(self,cityName):  cityDict = {**"北京"**: **"101010100"**,  **"上海"**: **"310000100"**,  **"天津"**: **"101030100"**}  **return** cityDict.get(cityName,**"101010100"**)   **def** clearText(self):  self.ui.textEdit.clear() **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app = QApplication(sys.argv)  dlg = MainDialog()  dlg.show()  sys.exit(app.exec\_()) |