

# 目录

简介	1.1
第一章 PySide6简介	1.2
第二章 窗体与控件	1.3
第三章 窗体与布局	1.4
第四章 无边框窗体	1.5
第五章 窗体间通信	1.6
第六章 国际化	1.7
第七章 打包	1.8

# PySide6 开发与实战

作者: iounce

## 阅读说明

本书是作者的一些经验总结，记录了PySide6开发图形界面应用程序中的常见要点，也包含一些示例讲解，希望能给大家一些启发，供参考之用。

## 第一章 PySide6简介

## 第二章 窗体与控件

## 第三章 窗体与布局

## 第四章 无边框窗体

## 第五章 窗体间通信

## 第六章 国际化

## 第七章 打包

# 第一章 PySide6简介

Qt是一款C++跨平台图形用户界面应用程序开发框架，完全面向对象，易扩展，实现了组件化编程。

PySide6是Qt的Python封装，支持Qt 6.0框架，其开源协议为[LGPLv3/GPLv2](#)，具体介绍可参考官网【[Qt for Python - Qt Wiki](#)】。

在创建基于组件的图形界面程序时，通常会用到以下三个模块：

- QtCore--提供基础功能，如信号和槽，属性系统，对象模型，文件系统，序列化等；
- QtGui--拓展了QtCore的图形界面功能，包括事件，窗口和屏幕，2D绘图及图像文本等；
- QtWidgets--提供了可用的组件，包括窗体及各类控件（按钮，文本框，输入框，菜单等）。

更多组件内容可参考官网的组件说明【[Qt for Python - Modules API](#)】。

接下来，将以Windows 10下的环境为例，来介绍PySide6的具体使用。

- 环境准备：
- 操作系统： Windows 10
- Python版本： 3.10.4
- 编辑器： VSCode
- 安装：
- 安装Python： 直接在【[Python官网](#)】下载指定版本3.10.4安装程序，然后双击安装即可。安装完成后，在命令行窗口输入Python，有如下结果即证明安装正确：

```
Python 3.10.4 (tags/v3.10.4:9d38120, Mar 23 2022, 23:13:41) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

- 安装PySide6： 直接在命令行输入`pip install pyside6`安装，等待安装完成即可。可通过`pip show pyside6`查看安装后的信息：

```
Name: PySide6
Version: 6.4.0.1
Summary: Python bindings for the Qt cross-platform application and UI framework
Home-page: https://www.pyside.org
Author: Qt for Python Team
Author-email: pyside@qt-project.org
License: LGPL
Location: d:\application\python\python310\lib\site-packages
Requires: PySide6-Addons, PySide6-Essentials, shiboken6
Required-by:
```

## 第二章 窗体与控件

PySide6中有各类丰富的控件，与Qt中的用法基本一致，只是带有Python语言的特性。常用控件包括窗体，菜单，对话框，输入框，文本框等等，一个完整的图形界面程序依赖于各类控件来实现。

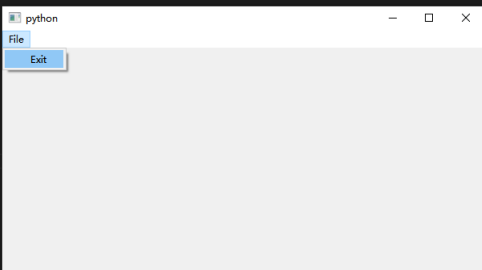
以下是一个简单的示例：

```
1 import sys
2 from PySide6 import QtWidgets
3
4 if __name__ == "__main__":
5     app = QtWidgets.QApplication([])
6
7     win = QtWidgets.QWidget()
8     win.resize(600, 300)
9     win.show()
10
11     sys.exit(app.exec())
```

首先需要引用PySide6包，然后使用其中的组件展示即可。

接下来实现一个自定义组件类，丰富界面的功能。

```
1 import sys
2 from PySide6 import QtWidgets, QtGui
3
4 class MyWindow(QtWidgets.QMainWindow):
5     def __init__(self):
6         super().__init__()
7
8         exit_action = QtGui.QAction('Exit', self)
9         exit_action.triggered.connect(self.close)
10
11         menubar = self.menuBar()
12         file_menu = menubar.addMenu('&File')
13         file_menu.addAction(exit_action)
14
15 if __name__ == "__main__":
16     app = QtWidgets.QApplication([])
17
18     win = MyWindow()
19     win.resize(600, 300)
20     win.show()
21
22     sys.exit(app.exec())
23
```

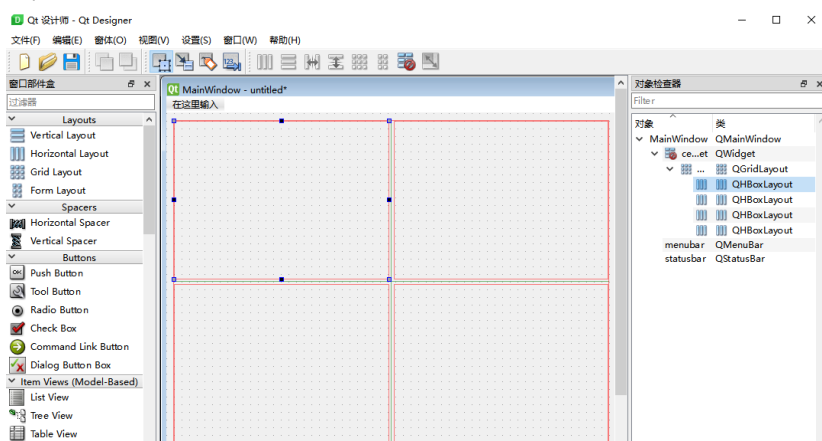


这里新建了自定义类MyWindow，继承自系统主窗体类QMainWindow，包含一些基本属性，在初始化函数init里面，在默认菜单栏增加了菜单项及其响应操作。

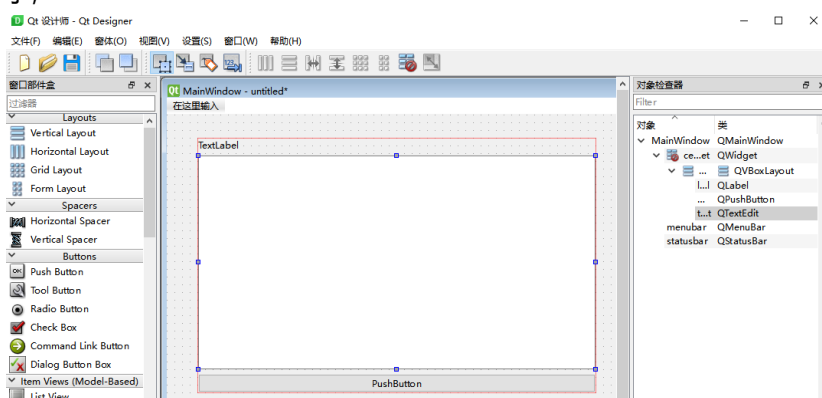
# 第三章 窗体与布局

PySide6中常见的布局控件有网格布局控件QGridLayout，垂直布局控件QVBoxLayout和水平布局控件QHBoxLayout。具体用法如下：

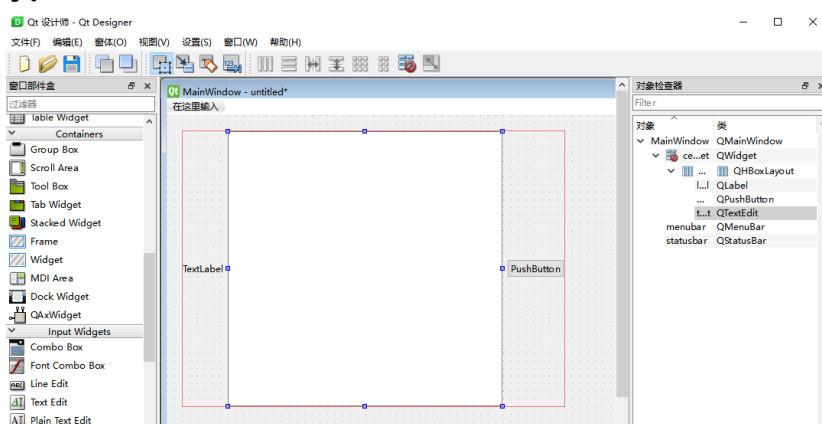
- QGridLayout：通常用于将空间分割为不同行列的单元格，可以自适应窗体大小；



- QVBoxLayout：通常用于将控件垂直对齐，即上下排列，可以自适应窗体大小；

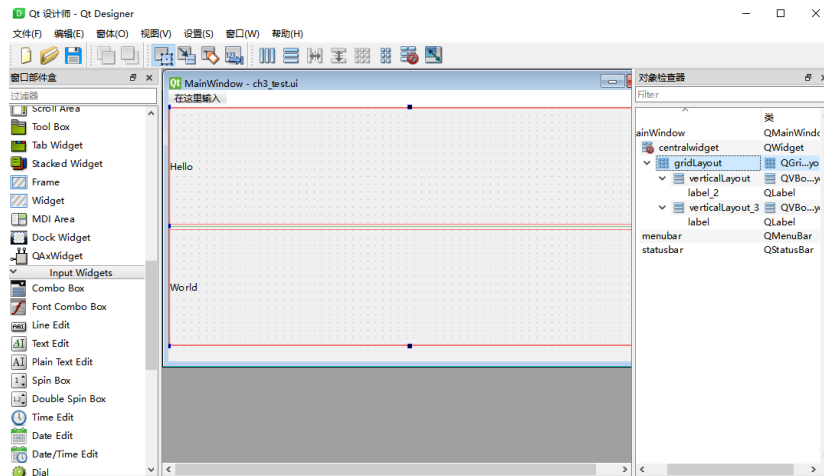


- QHBoxLayout：通常用于将控件水平对齐，即左右排列，可以自适应窗体大小。



对窗体控件进行布局，可以采用安装包提供的设计工具【Qt designer】来编辑，或者直接使用编码的方式来实现。

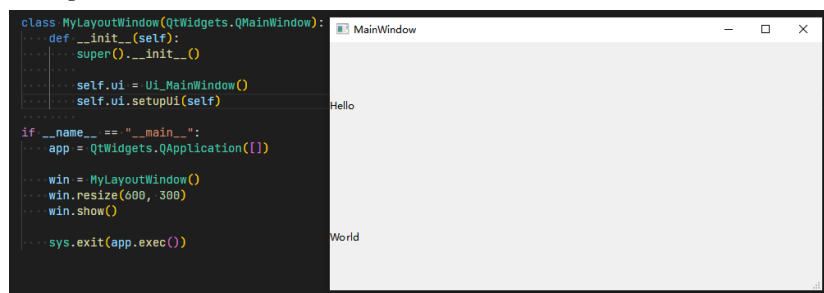
- 采用设计工具实现：打开设计工具，然后新建主窗体，增加一个网格布局控件QGridLayout，在其上面添加两个垂直布局控件QVBoxLayout，将界面分为上下两个部分。然后，分别在QVBoxLayout布局中添加QLabel控件。



- 直接调用ui文件：首先需要解析出ui文件中的窗体类和基类，然后实现自定义类即可：



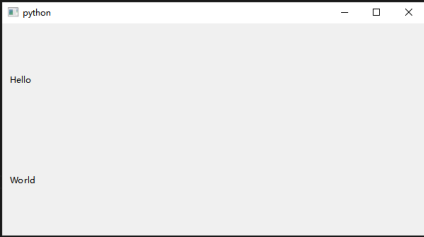
- 调用编译后的ui文件：首先需要使用 `pyside6-uic ch3_test.ui > ch3_test.py` 命令生成Python文件，然后引用界面类 `Ui_MainWindow` 即可：



- 采用直接编码实现：

编码实现首先需要明确具体的布局，然后再编码实现，相对于设计工具而言要求更高，但是更为灵活，且不需要依赖ui文件。

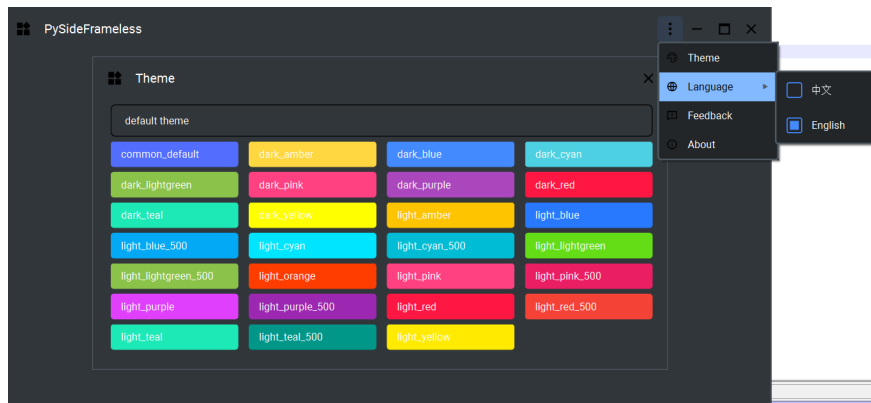
```
35 class MyLayoutWindow(QtWidgets.QWidget):
36     def __init__(self):
37         super().__init__()
38
39         self.grid_layout = QtWidgets.QGridLayout(self)
40         hello_layout = QtWidgets.QVBoxLayout()
41         world_layout = QtWidgets.QVBoxLayout()
42
43         self.grid_layout.addLayout(hello_layout, 0, 0, 1, 2)
44         self.grid_layout.addLayout(world_layout, 1, 0, 1, 2)
45
46         hello_label = QtWidgets.QLabel(self)
47         hello_label.setText("Hello")
48         hello_layout.addWidget(hello_label)
49
50         world_label = QtWidgets.QLabel(self)
51         world_label.setText("World")
52         world_layout.addWidget(world_label)
53
54 if __name__ == "__main__":
55     app = QtWidgets.QApplication([])
56
57     win = MyLayoutWindow()
58     win.resize(400, 300)
59     win.show()
60
61     sys.exit(app.exec())
62
```



说明： designer.exe工具可在Python安装目录找到， 参考路径： D:\Python\Python310\Lib\site-packages\PySide6\designer.exe pyside6-uic.exe工具可在Python安装目录找到， 参考路径： D:\Python\Python310\Scripts\pyside6-uic.exe







## 第五章 窗体间通信

PySide6中使用多个窗体的时候，常常会涉及到窗体间的数据传递，在Qt中常用的是信号与槽，PySide6中也有类似的处理机制，通常是使用信号Signal。

在一个窗体中定义类属性的Signal信号变量，在需要的地方调用其emit函数发送信号。然后，在另外一个窗体中调用信号的connect函数，绑定此信号的处理函数，这样就可以获取此前发送的数据。

示例如下：

```
77 class MyMainWindow(QtWidgets.QWidget):
78     def __init__(self):
79         super().__init__()
80
81         self.grid_layout = QtWidgets.QGridLayout(self)
82         self.hello_layout = QtWidgets.QVBoxLayout()
83         self.world_layout = QtWidgets.QVBoxLayout()
84
85         self.grid_layout.addLayout(self.hello_layout, 0, 0, 1, 2)
86         self.grid_layout.addLayout(self.world_layout, 1, 0, 1, 2)
87
88         self.hello_btn = QtWidgets.QPushButton(self)
89         self.hello_btn.setText("Click me")
90         self.hello_btn.clicked.connect(self.show_menu)
91         self.hello_layout.addWidget(self.hello_btn)
92
93         self.world_lbl = QtWidgets.QLabel(self)
94         self.world_layout.addWidget(self.world_lbl)
95
96         self.data_widget = self.world_lbl
97
98     def show_menu(self):
99         win = MySubWindow(self)
100         win.resize(300, 200)
101         win.data_signal.connect(self.on_signal)
102         win.show()
103
104     def on_signal(self, msg):
105         print(msg)
106         self.data_widget.setText(msg)

```

```
108 class MySubWindow(QtWidgets.QDialog):
109     data_signal = QtCore.Signal(str)
110
111     def __init__(self, parent):
112         super().__init__(parent)
113
114         self.signal_btn = QtWidgets.QPushButton(self)
115         self.signal_btn.setText("Send message")
116         self.signal_btn.clicked.connect(self.send_msg)
117         self.count = 0
118
119     def send_msg(self):
120         self.count += 1
121         self.data_signal.emit("Signal here: " + str(self.count))

```

```
124     if __name__ == "__main__":
125         app = QtWidgets.QApplication([])
126
127         qt_material.apply_stylesheet(app, 'dark_blue.xml')
128
129         win = MyMainWindow()
130         win.resize(600, 300)
131         win.show()
132
133         sys.exit(app.exec())
```

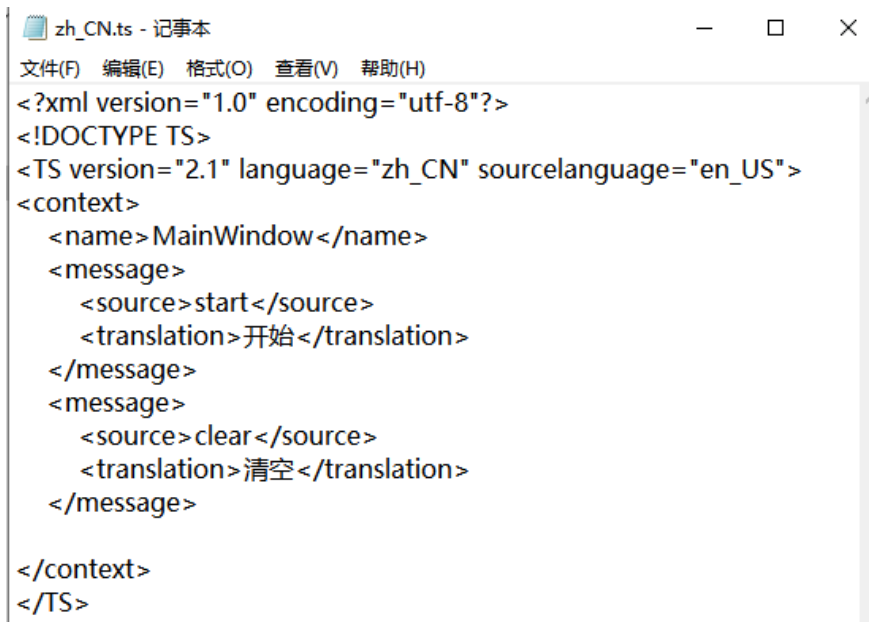
## 第六章 国际化

PySide6中使用窗体的时候，会广泛使用到各种文本信息，如果需要支持多语言的话，通常会比较复杂。

PySide6中可以使用语言工具【Qt Linguist】来转换不同的语言。

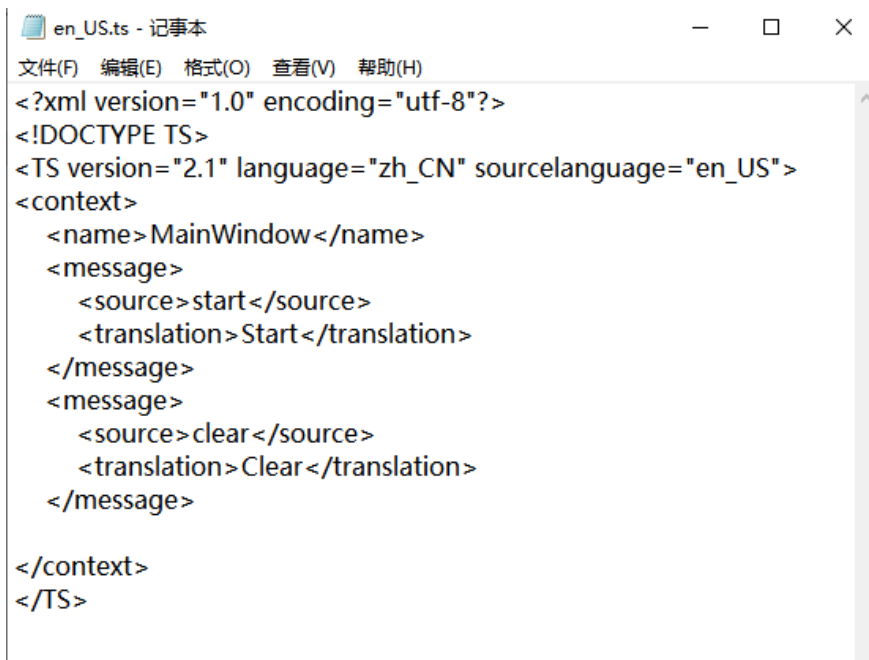
首先获取窗体文件的中文翻译源文件即ts文件，譬如，可使用命令`pyside6-lupdate main.py -ts zh_CN.ts`获取初步的翻译源文件。

然后可以手动编辑zh\_CN.ts文件，增加中文文本转换。示例如下：



```
zh_CN.ts - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS version="2.1" language="zh_CN" sourcelanguage="en_US">
<context>
  <name>MainWindow</name>
  <message>
    <source>start</source>
    <translation>开始</translation>
  </message>
  <message>
    <source>clear</source>
    <translation>清空</translation>
  </message>
</context>
</TS>
```

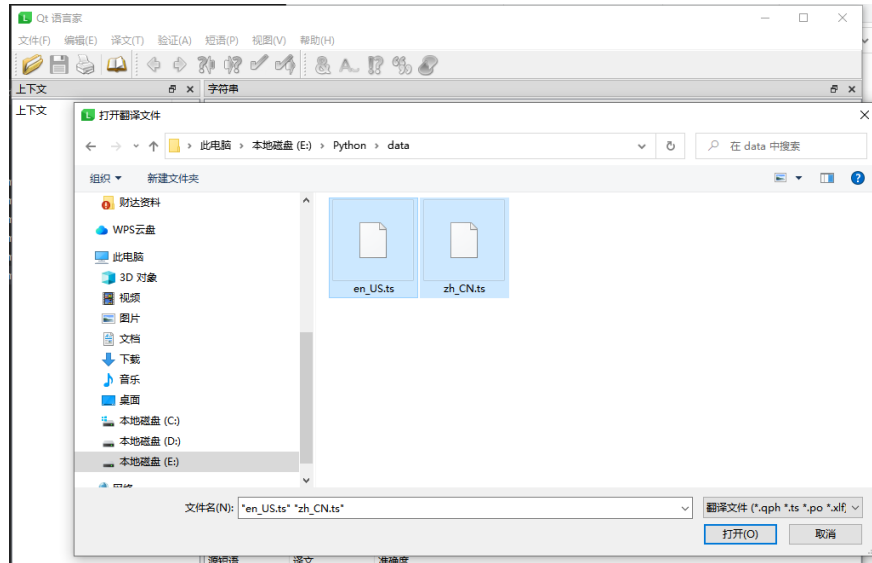
接下来，拷贝一份，修改文件为en\_US.ts，然后将中文手动翻译为英文。示例如下：



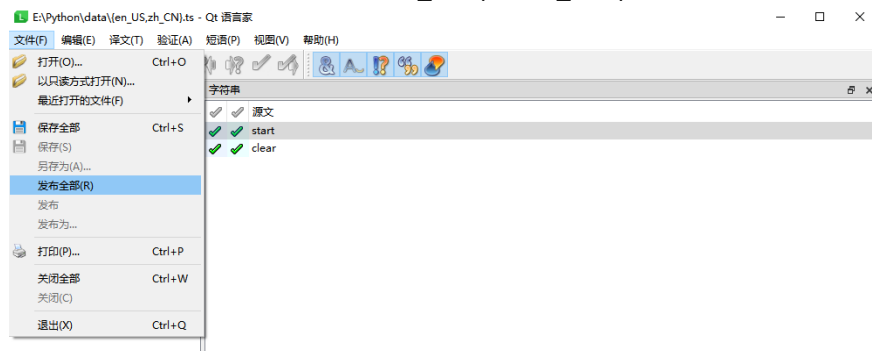
```
en_US.ts - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS version="2.1" language="zh_CN" sourcelanguage="en_US">
<context>
  <name>MainWindow</name>
  <message>
    <source>start</source>
    <translation>Start</translation>
  </message>
  <message>
    <source>clear</source>
    <translation>Clear</translation>
  </message>
</context>
</TS>
```

翻译源文件手动编辑完成后，就可以使用语言工具【Qt Linguist】将ts文件转换为目标翻译文件qm文件。

打开ts文件，可同时打开多个：



选择发布全部，就会在当前路径生成zh\_CN.qm和en\_US.qm文件：



最后在主程序中直接加载即可：

```
124 if __name__ == "__main__":
125     translator = QtCore.QTranslator()
126     translator.load('zh_CN')
127     ...
128     app = QtWidgets.QApplication([])
129     ...
130     app.installTranslator(translator)
131     ...
132     qt_material.apply_stylesheet(app, 'dark_blue.xml')
133     ...
134     win = MyMainWindow()
135     win.resize(600, 300)
136     win.show()
137     ...
138     sys.exit(app.exec())
139
```

如果要动态切换，则需要菜单中手动操作，切换为不同的目标语言。

说明： designer.exe工具可在Python安装目录找到，参考路径： D:\Python\Python310\Lib\site-packages\PySide6\linguist.exe pyside6-lupdate.exe工具可在Python安装目录找到，参考路径：

D:\Python\Python310\Scripts\pyside6-lupdate.exe

## 第七章 打包

PySide6窗体程序打包，一般使用Pyinstaller打包工具，它可以很方便的将Python脚本打包为可执行程序。

下面以Windows下操作为例，来作简单介绍。

首先使用命令`pip install pyinstaller`安装Pyinstaller工具，可用`pip show pyinstaller`查看安装结果：

```
Name: pyinstaller
Version: 5.1
Summary: PyInstaller bundles a Python application and all its dependencies into a single package.
Home-page: https://www.pyinstaller.org/
Author: Hartmut Goebel, Giovanni Bajo, David Viera, David Cortesi, Martin Zibricky
Author-email:
License: GPLv2-or-later with a special exception which allows to use PyInstaller to build and distribute non-free programs (including commercial ones)
Location: d:\application\python\python310\lib\site-packages
Requires: altgraph, pefile, pyinstaller-hooks-contrib, pywin32-ctypes, setuptools
Required-by: auto-py-to-exe
```

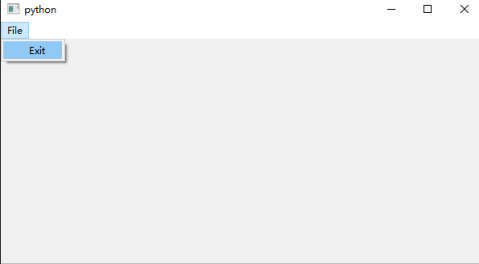
Pyinstaller工具的常用参数如下：

- -F：打包Python程序为单个可执行文件；
- -D：打包Python程序为一个文件夹；
- -i：生成图标，只适用于Windows平台；
- -n：指定打包后生成文件的名称；
- -w：禁止命令行对话框弹出。

更多其他参数，可参考Pyinstaller工具相关文档。

接下来，将如下PySide6窗体程序打包：

```
1 import sys
2 from PySide6 import QtWidgets, QtGui, QtCore
3
4 class MyWindow(QtWidgets.QMainWindow):
5     def __init__(self):
6         super().__init__()
7
8         self.exit_action = QtGui.QAction('&Exit', self)
9         self.exit_action.triggered.connect(self.close)
10
11         menubar = self.menuBar()
12         file_menu = menubar.addMenu('&File')
13         file_menu.addAction(self.exit_action)
14
15
16 if __name__ == '__main__':
17     app = QtWidgets.QApplication([])
18
19     win = MyWindow()
20     win.resize(600, 300)
21     win.show()
22
23     sys.exit(app.exec())
```



直接打包为单一可执行文件，执行如下操作：`Pyinstaller -F -w main.py -p D:\Application\Python\Python310\Lib\site-packages`

```
E:\Python\test>Pyinstaller -F -w main.py -p D:\Application\Python\Python310\Lib\site-packages
436 INFO: PyInstaller: 5.1
436 INFO: Python: 3.10.4
436 INFO: Platform: Windows-10-10.0.19042-SP0
457 INFO: wrote E:\Python\test\main.spec
490 INFO: UPX is available.
491 INFO: Extending PYTHONPATH with paths
['E:\Python\test', 'D:\Application\Python\Python310\Lib\site-packages']
22 WARNING: qt_material must be imported after PySide or PyQt!
pygame 2.1.2 (SDL 2.0.18, Python 3.10.4)
Hello from the pygame community. https://www.pygame.org/contribute.html
167 WARNING: discover_hook_directories: Failed to process hook entry point 'hook_dirs = playwright._impl._pyinstaller:get_hook_dirs': pkg_resources.VersionConflict: (greenlet 2.0.0 (d:\application\python\python310\lib\site-packages), Requirement.parse('greenlet==1.1.3'))
2051 INFO: checking Analysis
2062 INFO: checking PYZ
2067 INFO: checking PKG
2072 INFO: Bootloader D:\Application\Python\Python310\lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
2072 INFO: checking EXE
```

此处加了PySide6的安装路径，便于打包程序找到相关依赖文件。

执行命令后，生成文件如下：

此电脑 > 本地磁盘 (E:) > Python > test					在 test 中搜索	
名称	修改日期	类型	大小			
build	2023/6/5 17:51	文件夹				
dist	2023/6/5 17:59	文件夹				
main.py	2023/6/5 17:58	PY 文件	5 KB			
main.spec	2023/6/5 18:13	SPEC 文件	1 KB			

其中main.spec为打包配置文件，可手动修改：

```
main.spec - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
# -*- mode: python ; coding: utf-8 -*-

block_cipher = None

a = Analysis(
    ['main.py'],
    pathex=['D:\\Application\\Python\\Python310\\Lib\\site-packages'],
    binaries=[],
    datas=[],
    hiddenimports=[],
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
    noarchive=False,
)
pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)

exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.zipfiles,
    a.datas,
    [],
    name='main',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
```



build目录为编译生成的中间文件：

此电脑 > 本地磁盘 (E:) > Python > test > build > main					在 main 中搜索
名称	修改日期	类型	大小		
Analysis-00.toc	2023/6/5 17:59	TOC 文件	28 KB		
base_library.zip	2023/6/5 17:59	WinRAR ZIP 压缩...	813 KB		
EXE-00.toc	2023/6/5 17:59	TOC 文件	20 KB		
main.exe.manifest	2023/6/5 17:59	MANIFEST 文件	2 KB		
main.pkg	2023/6/5 17:59	PKG 文件	30,738 KB		
PKG-00.toc	2023/6/5 17:59	TOC 文件	19 KB		
PYZ-00.pyz	2023/6/5 17:59	Python Zip Appli...	814 KB		
PYZ-00.toc	2023/6/5 17:59	TOC 文件	10 KB		
Tree-00.toc	2023/6/5 17:51	TOC 文件	102 KB		
Tree-01.toc	2023/6/5 17:51	TOC 文件	10 KB		
Tree-02.toc	2023/6/5 17:51	TOC 文件	1 KB		
warn-main.txt	2023/6/5 17:59	TXT 文件	3 KB		
xref-main.html	2023/6/5 17:59	Microsoft Edge ...	297 KB		

dist目录才是最终可执行文件目录：

此电脑 > 本地磁盘 (E:) > Python > test > dist					在 dist 中搜索
名称	修改日期	类型	大小		
main.exe	2023/6/5 17:59	应用程序	31,051 KB		

此时，双击dist目录下的main.exe即可打开最终打包的程序。

如果将参数-F改为-D，执行命令：`Pyinstaller -D -w main.py -p`

`D:\Application\Python\Python310\Lib\site-packages`

```
E:\Python\test>Pyinstaller -D -w main.py -p D:\Application\Python\Python310\Lib\site-packages
421 INFO: PyInstaller: 5.1
422 INFO: Python: 3.10.4
428 INFO: Platform: Windows-10-10.0.19042-SP0
428 INFO: wrote E:\Python\test\main.spec
446 INFO: UPX is available.
447 INFO: Extending PYTHONPATH with paths
['E:\Python\test', 'D:\Application\Python\Python310\Lib\site-packages']
15 WARNING: qt_material must be imported after PySide or PyQt!
pygame 2.1.2 (SDL 2.0.18, Python 3.10.4)
Hello from the pygame community. https://www.pygame.org/contribute.html
127 WARNING: discover_hook_directories: Failed to process hook entry point 'hook-dirs = playwright_impl...pyinstaller:
et_hook_dirs': pkg_resources.VersionConflict: (greenlet 2.0.0 (d:\application\python\python310\lib\site-packages), Requ
rement.parse('greenlet==1.1.3'))
1635 INFO: checking Analysis
1646 INFO: checking PYZ
1651 INFO: checking PKG
1652 INFO: Bootloader D:\Application\Python\Python310\Lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
1652 INFO: checking EXE
1653 INFO: checking COLLECT
```

这个时候，除了main.exe，还会生成其他多个依赖文件，具体如下：

此电脑 > 本地磁盘 (E:) > Python > test > dist > main >

名称	修改日期	类型	大小
Qt6Gui.dll	2022/12/28 18:01	应用程序扩展	7,475 KB
Qt6Widgets.dll	2022/12/28 18:01	应用程序扩展	5,898 KB
Qt6Core.dll	2022/12/28 18:01	应用程序扩展	5,555 KB
opengl32sw.dll	2022/12/28 18:01	应用程序扩展	5,446 KB
Qt6Quick.dll	2022/12/28 18:01	应用程序扩展	5,019 KB
Qt6Qml.dll	2022/12/28 18:01	应用程序扩展	4,389 KB
Qt6OpenGL.dll	2022/12/28 18:01	应用程序扩展	1,880 KB
python310.dll	2023/1/4 16:03	应用程序扩展	1,466 KB
Qt6Network.dll	2022/12/28 18:01	应用程序扩展	1,375 KB
main.exe	2023/6/5 18:21	应用程序	1,145 KB
libcrypto-1_1.dll	2022/12/28 18:01	应用程序扩展	1,080 KB
ucrtbase.dll	2023/6/5 11:35	应用程序扩展	1,012 KB
base.library.zip	2023/6/5 18:21	WinRAR ZIP 压缩...	813 KB
Qt6QmlModels.dll	2022/12/28 18:01	应用程序扩展	659 KB
MSVCP140.dll	2023/1/4 16:03	应用程序扩展	551 KB
Qt6VirtualKeyboard.dll	2022/12/28 18:01	应用程序扩展	434 KB
Qt6Svg.dll	2022/12/28 18:01	应用程序扩展	351 KB
unicodedata.pyd	2023/1/4 16:03	Python Extensio...	286 KB
MSVCP140_2.dll	2023/1/4 16:03	应用程序扩展	182 KB
shiboken6.abi3.dll	2022/12/28 18:01	应用程序扩展	131 KB
_decimal.pyd	2023/1/4 16:03	Python Extensio...	102 KB
VCRUNTIME140.dll	2023/1/4 16:03	应用程序扩展	95 KB
pyside6.abi3.dll	2022/12/28 18:01	应用程序扩展	89 KB
_lzma.pyd	2023/1/4 16:03	Python Extensio...	82 KB
python3.dll	2023/1/4 16:03	应用程序扩展	61 KB
_bz2.pyd	2023/1/4 16:03	Python Extensio...	44 KB
_socket.pyd	2023/1/4 16:03	Python Extensio...	39 KB
VCRUNTIME140_1.dll	2023/1/4 16:03	应用程序扩展	37 KB
_hashlib.pyd	2023/1/4 16:03	Python Extensio...	32 KB
MSVCP140_1.dll	2023/1/4 16:03	应用程序扩展	24 KB
select.pyd	2023/1/4 16:03	Python Extensio...	22 KB
api-ms-win-core-file-l1-1-0.dll	2023/6/5 11:36	应用程序扩展	22 KB
api-ms-win-core-localization-l1-2-0.dll	2023/6/5 11:35	应用程序扩展	21 KB
api-ms-win-crt-math-l1-1-0.dll	2023/6/5 11:35	应用程序扩展	21 KB
api-ms-win-core-processthreads-l1-1-0.dll	2023/6/5 11:35	应用程序扩展	21 KB
api-ms-win-core-synch-l1-1-0.dll	2023/6/5 11:36	应用程序扩展	21 KB
api-ms-win-core-processenvironmen...	2023/6/5 11:36	应用程序扩展	20 KB