第 3 章

wxPython

3.1 wxPython 的安装

由于 wxPython 不是 Python 官方提供的图形用户界面开发工具包，所以在使用 wxPython 之前，需要安装 wxPython。

安装 wxPython 的方法很简单，打开“命令提示符”，并输入命令 pip install wxpython 即可。

在完成安装之后，需要引入该包才可以正常使用 wxPython 进行编程，需要注意的是，引入的包名是 wx，而不是 wxpython，示例代码如下：

```
#资源包\Code\chapter3\3.1\0301.py
import wx
```

3.2 wxPython 的基本要素

在正式开始学习 wxPython 之前，首先要了解一下 wxPython 的 5 个基本要素，其分别为应用程序、窗口、控件、布局管理器和事件处理，之后的学习内容也都是紧紧围绕这 5 个基本要素展开的。

3.3 应用程序

在 wxPython 中，应用程序主要用于管理主事件循环，主事件循环是 wxPython 驱动程序的动力，如果没有应用程序，wxPython 的程序将无法正常运行。

1. 创建应用程序对象

可以通过 wx 模块中的 App 类创建应用程序对象，用于完成应用程序的创建，其语法格式如下：

```
App()
```

2. 应用程序对象的相关方法

应用程序对象的相关方法为 MainLoop()方法，主要用于主事件循环，其语法格式如下：

```
MainLoop()
```

3. 创建应用程序

创建应用程序有两种方式，分别为使用 App 类和 App 类的子类。

1) 使用 App 类创建应用程序

示例代码如下：

```
#资源包\Code\chapter3\3.3\0302.py
import wx
#创建应用程序
app = wx.App()
#主事件循环
app.MainLoop()
```

2) 使用 App 类的子类创建应用程序

需要注意的是，App 类中具有两种方法，分别为 OnInit()方法和 OnExit()方法，App 类的子类必须重写这两种方法，其中，OnInit()方法表示应用程序进入时所调用的方法，而 OnExit()方法表示应用程序退出时所调用的方法，示例代码如下：

```
#资源包\Code\chapter3\3.3\0303.py
import wx
class App(wx.App):
    def OnInit(self):
        return True
    def OnExit(self):
        return False
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

3.4 窗口

可以通过 Window 类的子类来创建各种窗口，包括框架(Frame 类)、内容面板(Panel 类)、菜单栏(MenuBar 类)和分隔窗口(SplitterWindow 类)等。

3.4.1 框架(Frame 类)

在 wxPython 中，框架主要用于管理窗体的相关控件，并呈现给用户。

1. 创建框架对象

可以通过 wx 模块中的 Frame 类创建框架对象，用于完成框架的创建，其语法格式

如下：

```
Frame(parent, id, title, pos, size, style, name)
```

其中，参数 parent 表示框架的父窗口，该值一般为 None；参数 id 表示窗口标识符；参数 title 表示框架的标题；参数 pos 表示框架的位置；参数 size 表示框架的尺寸；参数 style 表示框架的类型；参数 name 表示框架的名称。

在上面的参数中有一个重要的概念，即窗口标识符，窗口标识符指的是在事件中用于唯一确定窗口的标识，该标识既可以为常量，也可以为该常量的值。

创建窗口标识符有以下 3 种方法。

1）自动创建窗口标识符

通过常量 ID_ANY，或者其对应的值－1，即可使 wxPython 自动创建窗口标识符。通常情况下，在不需要修改子窗口或控件状态的时候，一般让 wxPython 自动创建窗口标识符，例如创建一个不需要改变的静态文本控件。

2）使用标准窗口标识符

wxPython 中提供了标准窗口标识符，这些窗口标识符的范围在常量 ID_LOWEST 和常量 ID_HIGHEST 之间，即 4999～5999，常用的窗口标识符如表 3-1 所示。

表 3-1　常用的窗口标识符

常　　量	值	描　　述
ID_OPEN	5000	打开(O)...
ID_CLOSE	5001	关闭
ID_NEW	5002	新建(N)
ID_SAVE	5003	保存(S)
ID_SAVEAS	5004	另存为(A)...
ID_EXIT	5006	退出(Q)
ID_UNDO	5007	撤销(U)
ID_REDO	5008	恢复(R)
ID_HELP	5009	帮助(H)
ID_PRINT	5010	打印(P)...
ID_PREVIEW	5013	打印预览(W)...
ID_ABOUT	5014	关于(A)
ID_EDIT	5030	编辑(E)
ID_CUT	5031	剪切(t)
ID_COPY	5032	复制(C)
ID_PASTE	5033	粘贴(P)
ID_CLEAR	5034	清除(C)
ID_FIND	5035	Find...
ID_SELECTALL	5037	全部选择(A)
ID_DELETE	5038	删除(D)
ID_REPLACE	5039	Replace...
ID_PROPERTIES	5041	属性(P)
ID_FILE	5050	文件(F)

续表

常　量	值	描　述
ID_OK	5100	确认
ID_CANCEL	5101	取消
ID_APPLY	5102	应用(A)
ID_YES	5103	是(Y)
ID_NO	5104	否(N)
ID_FORWARD	5106	前进(F)
ID_BACKWARD	5107	返回(B)
ID_UP	5120	向上(U)
ID_DOWN	5121	向下(D)
ID_HOME	5122	Home(H)
ID_REFRESH	5123	刷新
ID_STOP	5124	停止(S)
ID_INDEX	5125	索引(I)
ID_BOLD	5126	粗体(B)
ID_ITALIC	5127	斜体(I)
ID_JUSTIFY_CENTER	5128	居中
ID_JUSTIFY_FILL	5129	分散对齐
ID_JUSTIFY_RIGHT	5130	右对齐
ID_JUSTIFY_LEFT	5131	左对齐
ID_UNDERLINE	5132	下画线
ID_INDENT	5133	缩进
ID_UNINDENT	5134	取消缩进(U)
ID_ZOOM_100	5135	实际大小(A)
ID_ZOOM_FIT	5136	缩放以适应窗口(F)
ID_ZOOM_IN	5137	放大(I)
ID_ZOOM_OUT	5138	缩小(O)
ID_UNDELETE	5139	取消删除
ID_REVERT_TO_SAVED	5140	还原为上次保存的文件

3) 自定义窗口标识符

自定义窗口标识符必须是正值,并且要避免在窗口标识符常量所在的范围之内自定义窗口标识符。

2. 框架对象的相关方法

1) Show()方法

该方法从 Window 类中继承而来,用于显示窗口,其语法格式如下:

```
Show()
```

2) Close()方法

该方法从 Window 类中继承而来,用于关闭窗口,其语法格式如下:

```
Close()
```

3）Destroy()方法

该方法从 Window 类中继承而来，用于销毁窗口，其语法格式如下：

```
Destroy()
```

4）SetTitle()方法

该方法从 TopLevelWindow 类中继承而来，用于设置窗口标题，其语法格式如下：

```
SetTitle(title)
```

其中，参数 title 表示窗口的标题。

5）SetIcon()方法

该方法从 TopLevelWindow 类中继承而来，用于设置窗口图标，其语法格式如下：

```
SetIcon(icon)
```

其中，参数 icon 表示窗口图标。

6）SetSizeHints()方法

该方法从 TopLevelWindow 类中继承而来，用于设置窗口的最小尺寸和最大尺寸，其语法格式如下：

```
SetSizeHints(minSize, maxSize)
```

其中，参数 minSize 表示最小尺寸；参数 maxSize 表示最大尺寸。

7）SetMenuBar()方法

该方法用于将菜单栏添加到框架中，其语法格式如下：

```
SetMenuBar(menuBar)
```

其中，参数 menuBar 表示菜单栏对象。

8）SetToolBar()方法

该方法用于将工具栏添加到框架中，其语法格式如下：

```
SetToolBar(toolBar)
```

其中，参数 toolBar 表示工具栏对象。

9）SetStatusBar()方法

该方法用于将状态栏添加到框架中，其语法格式如下：

```
SetStatusBar(statusBar)
```

其中,参数 statusBar 表示状态栏对象。

3. 创建框架

创建框架有两种方式,分别为使用 Frame 类和 Frame 类的子类。

1) 使用 Frame 类创建框架

示例代码如下:

```
#资源包\Code\chapter3\3.4\0304.py
import wx
class App(wx.App):
    def OnInit(self):
        frame = wx.Frame(parent=None, title='框架(Frame类)', pos=(100, 100),
size=(600, 500))
        frame.Show()
        return True
    def OnExit(self):
        print('应用程序已退出,谢谢使用!')
        return False
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

上面代码的运行结果如图 3-1 所示。

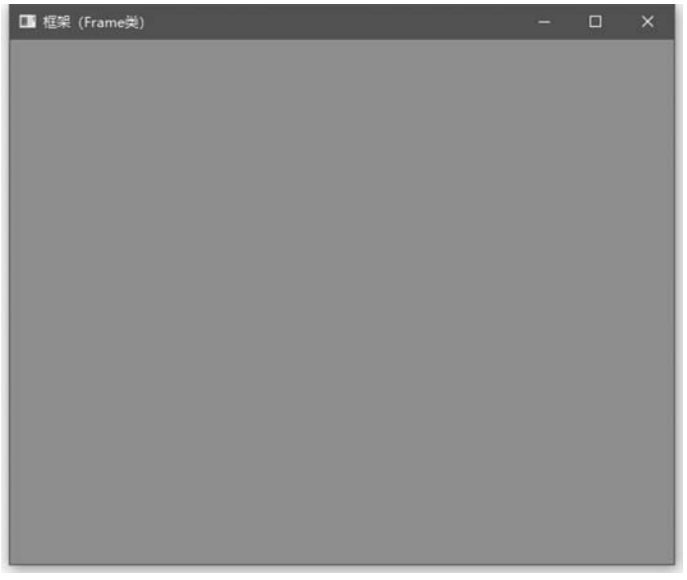

图 3-1 框架(Frame 类)

2) 使用 Frame 类的子类创建框架

示例代码如下:

```
#资源包\Code\chapter3\3.4\0305.py
import wx
```

```
class MyFrame(wx.Frame):
    def __init__(self):
        super().__init__(parent = None, title = '框架(Frame类)', pos = (100, 100),
size = (600, 500))
class App(wx.App):
    def OnInit(self):
        frame = MyFrame()
        frame.Show()
        return True
    def OnExit(self):
        print('应用程序已退出,谢谢使用!')
        return False
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

上面代码的运行结果如图 3-2 所示。

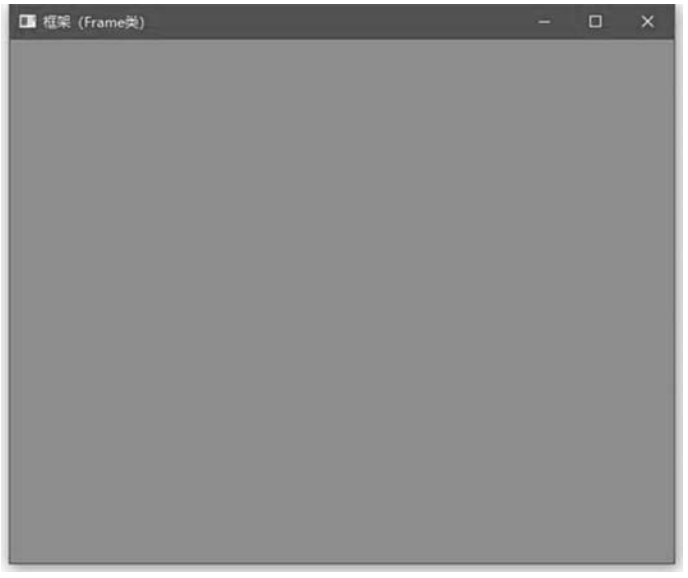

图 3-2　框架(Frame 类)

3.4.2　内容面板(Panel 类)

在 wxPython 中,内容面板是一个容器元素,可以在其上添加控件。

1. 创建内容面板对象

可以通过 wx 模块中的 Panel 类创建内容面板对象,用于完成内容面板的创建,其语法如下:

```
Panel(parent, id, pos, size, style, name)
```

其中,参数 parent 表示内容面板的父窗口;参数 id 表示窗口标识符;参数 pos 表示内

容面板的位置；参数 size 表示内容面板的尺寸；参数 style 表示内容面板的类型；参数 name 表示内容面板的名称。

2. 创建内容面板

示例代码如下：

```
#资源包\Code\chapter3\3.4\0306.py
import wx
class MyFrame(wx.Frame):
    def __init__(self):
        super().__init__(parent = None, title = '内容面板(Panel 类)', pos = (100, 100), size
= (600, 500))
        #内容面板的父窗口为框架
        panel = wx.Panel(parent = self)
class App(wx.App):
    def OnInit(self):
        frame = MyFrame()
        frame.Show()
        return True
    def OnExit(self):
        print('应用程序已退出,谢谢使用!')
        return False
if __name__ == '__main__':
    app = App()
    app.MainLoop()
```

上面的代码的运行结果如图 3-3 所示。

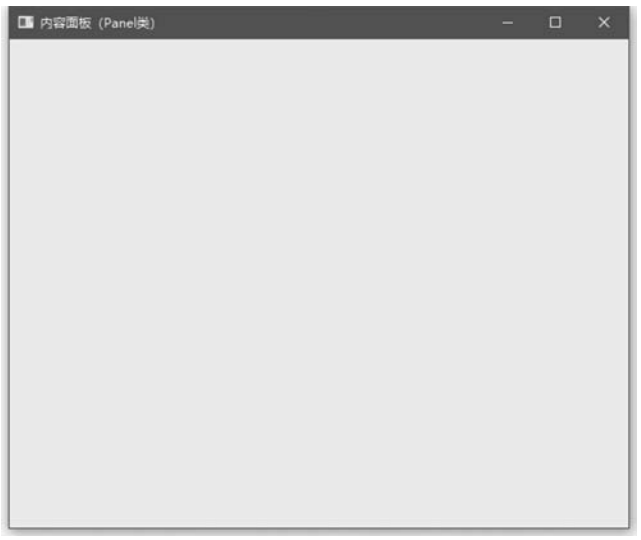

图 3-3 内容面板(Panel 类)

3.4.3 菜单栏(MenuBar 类)

菜单栏实际上是一种树形结构，为软件的大多数功能提供功能入口。在 wxPython 中，

菜单栏中仅可以包含菜单，而在菜单中则可以包含菜单和菜单项，其布局如图 3-4 所示。

图 3-4　菜单栏的布局

1. 创建菜单栏对象

可以通过 wx 模块中的 MenuBar 类创建菜单栏对象，用于完成菜单栏的创建，其语法格式如下：

```
MenuBar()
```

2. 菜单栏对象的相关方法

菜单栏对象的相关方法为 Append()方法，主要用于将菜单添加至菜单栏中，其语法格式如下：

```
Append(menu, item)
```

其中，参数 menu 表示要添加的菜单对象；参数 item 表示要添加菜单的名称。

3. 创建菜单对象

可以通过 wx 模块中的 Menu 类创建菜单对象，用于完成菜单的创建，其语法格式如下：

```
Menu()
```

4. 菜单对象的相关方法

1）Append()方法

该方法用于将菜单或菜单项添加至菜单中，其语法格式有以下两种：

```
Append(id, item, menu)
```

其中，参数 id 表示窗口标识符；参数 item 表示要添加菜单的名称；参数 menu 表示要添加的菜单对象。

```
Append(menuitem)
```

其中，参数 menuitem 表示要添加的菜单项对象。

2）AppendCheckItem()方法

该方法用于添加可选中菜单项，其语法格式如下：

```
AppendCheckItem(id, item)
```

其中，参数 id 表示窗口标识符；参数 item 表示要添加菜单项的名称。

3）AppendSeparator()方法

该方法用于添加分隔符，其语法格式如下：

```
AppendSeparator()
```

4）FindItemById()方法

该方法用于通过窗口标识符查找菜单项，其语法格式如下：

```
FindItemById(id)
```

其中，参数 id 表示窗口标识符。

5. 创建菜单项对象

可以通过 wx 模块中的 MenuItem 类创建菜单项对象，用于完成菜单项的创建，其语法格式如下：

```
MenuItem(parentMenu, id, text, helpString, kind)
```

其中，参数 parentMenu 表示菜单项对象的父菜单；参数 id 表示窗口标识符；参数 text 表示菜单项的名称；参数 helpString 用于在状态栏中显示提示信息；kind 表示菜单项类型，包括 wx.ITEM_NORMAL(普通菜单项，默认)、wx.ITEM_SEPARATOR(分割线菜单项)、wx.ITEM_CHECK(复选框菜单项)和 wx.ITEM_RADIO(单选按钮菜单项)。

6. 菜单项对象的相关方法

1）GetItemLabel()方法

该方法用于获取菜单项的标签内容，其语法格式如下：

```
GetItemLabel()
```

2）GetItemLabelText()方法

该方法同样用于获取菜单项的标签内容，但与 GetItemLabel()方法不同的是，该方法不会解析特殊字符，其语法格式如下：