Using bbfreeze's Advanced Configuration

WE'LL COVER THE FOLLOWING

- Wrapping Up

The PyPI page for bbfreeze (which is also its home page) has very little documentation. However, the page does say that the preferred way to use bbfreeze is with little scripts. We're going to try creating a binary with the wxPython example, mentioned earlier. Here's the wx code:

```
import wx

class DemoPanel(wx.Panel):
    """"""

    def __init__(self, parent):
        """Constructor"""
        wx.Panel.__init__(self, parent)

        labels = ["Name", "Address", "City", "State", "Zip",
                  "Phone", "Email", "Notes"]

        mainSizer = wx.BoxSizer(wx.VERTICAL)
        lbl = wx.StaticText(self, label="Please enter your information here:")
        lbl.SetFont(wx.Font(12, wx.SWISS, wx.NORMAL, wx.BOLD))
        mainSizer.Add(lbl, 0, wx.ALL, 5)
        for lbl in labels:
            sizer = self.buildControls(lbl)
            mainSizer.Add(sizer, 1, wx.EXPAND)
        self.SetSizer(mainSizer)
        mainSizer.Layout()

    def buildControls(self, label):
        """
        Put the widgets together
        """
        sizer = wx.BoxSizer(wx.HORIZONTAL)
        size = (80,40)
        font = wx.Font(12, wx.SWISS, wx.NORMAL, wx.BOLD)

        lbl = wx.StaticText(self, label=label, size=size)
        lbl.SetFont(font)
        sizer.Add(lbl, 0, wx.ALL|wx.CENTER, 5)
```

```
        if label != "Notes":
            txt = wx.TextCtrl(self, name=label)
        else:

            txt = wx.TextCtrl(self, style=wx.TE_MULTILINE, name=label)
        sizer.Add(txt, 1, wx.ALL, 5)
        return sizer

class DemoFrame(wx.Frame):
    """
    Frame that holds all other widgets
    """

    def __init__(self):
        """Constructor"""
        wx.Frame.__init__(self, None, wx.ID_ANY,
                          "Py2Exe Tutorial",
                          size=(600,400)
                          )
        panel = DemoPanel(self)
        self.Show()

if __name__ == "__main__":
    app = wx.App(False)
    frame = DemoFrame()
    app.MainLoop()
```

Now let's create a simple freezing script!

```
# bb_setup.py
from bbfreeze import Freezer

f = Freezer(distdir="bb-binary")
f.addScript("sampleApp.py")
f()
```

First off, we import the **Freezer** class from the **bbfreeze** package. Freezer accepts three arguments: a destination folder, an **includes** iterable and an **excludes** iterable (i.e. a tuple or list). Just to see how well bbfreeze works with only its defaults, we leave out the includes and excludes tuples/lists. Once you have a Freezer object, you can add your script(s) by calling the Freezer object name's addScript method. Then you just need to call the object (i.e. **f()**).

Note: You may see a warning about bb_freeze not being able to find "MSVCP90.dll" or similar. If you see that message, you may need to include it explicitly or add it as a dependency when you create an installer. We will be learning about how to create an installer in a later chapter.

To run this script, you just have to do something like this:

```
python bb_setup.py
```

When I ran this script, it created a folder named **bb-binary** that contained 19 files that weighed in at 17.2 MB. When I ran the **sampleApp.exe** file, it ran just fine and was properly themed, however it also had a console screen. We'll have to edit our script a bit to fix that:

```
# bb_setup2.py
from bbfreeze import Freezer

includes = []
excludes = ['_gtkagg', '_tkagg', 'bsddb', 'curses', 'email', 'pywin.debugger',
            'pywin.debugger.dbgcon', 'pywin.dialogs', 'tcl',
            'Tkconstants', 'Tkinter']

bbFreeze_Class = Freezer('dist', includes=includes, excludes=excludes)

bbFreeze_Class.addScript("sampleApp.py", gui_only=True)

bbFreeze_Class.use_compression = 0
bbFreeze_Class.include_py = True
bbFreeze_Class()
```

If you run this, you should end up with a **dist** folder with about 19 files, but a slightly different size of 19.6 MB. Notice that we added a second argument to the addScript method: gui_only=True. This makes that annoying console go away. We also set compression to zero (no compression) and include the Python interpreter. Turning on compression only reduced the result back down to 17.2 MB though.

The bbfreeze package also handles "recipes" and includes several examples, however they are not documented well either. Feel free to study them yourself as an exercise.

Wrapping Up

Now you should know the basics of using bbfreeze to create binaries from your programs. I noticed that when I ran bbfreeze on my machine, it was considerably slower in producing the wxPython executable compared with py2exe. This is one of those things that you will have to experiment with when you are determining which tool to use to create your binaries.