Tutorial

py2exe turns Python programs into packages that can be run on other Windows computers without needing to install Python on those computers. Python is needed on the computer where py2exe itself is run because py2exe is a Python program and it includes parts of Python in the package that is built.

To successfully complete this tutorial you'll need to know the basics of Python (you can get started at python.org's 🌐 getting started page). You'll also need to know 🌐 how to run Python programs from the command prompt.

There are a few simple steps needed to use py2exe once you've installed it:

1. Create/test your program
2. Create your setup script (setup.py)
3. Run your setup script
4. Test your executable
5. Providing the Microsoft Visual C runtime DLL
     5.1. Python 2.4 or 2.5
     5.2. Python 2.6, 2.7, 3.0, 3.1
         5.2.1. Bundling the C runtime DLL
             5.2.1.1 win32ui special case
         5.2.2. Running the redistributable C runtime installer
6. Build an installer if applicable

1. Create/test your program

The biggest step is almost always the first one. The good news is that py2exe typically has little or no impact on this step. The vast majority of things you can do with Python will work with py2exe. Many modules just work seamlessly with py2exe, but some third party modules will require a little extra work. Luckily there is help available at WorkingWithVariousPackagesAndModules.

It's important that you make sure everything is working before you use py2exe. If py2exe fixes a broken program, then that's probably a bug in py2exe that needs to be fixed!

The first example we'll use here is our old friend...

```
[des]activar nros. de línea

    1 print "Hello World!"
```

📎 hello.py

We need to make sure it's working...

```
C:\Tutorial>python hello.py
Hello World!

C:\Tutorial>
```

Looks good!

2. Create your setup script (setup.py)

py2exe extends 🌐 Distutils with a new "command". If you've installed third party Python modules then there's a good chance you've seen at least one distutils command:

```
C:\Tutorial>python setup.py install
```

"install" is a Distutils command that installs something (typically a Python module or package). The details Distutils needs to do that installation are contained in setup.py (and sometimes other associated files).

"py2exe" is a new Distutils command that is added when you import py2exe. To use py2exe you need to create a setup.py file to tell Distutils and py2exe what you want to do. Here's a setup.py whose simplicity is appropriate for our sample program...

```
[des]activar nros. de línea

1 from distutils.core import setup
2 import py2exe
3
4 setup(console=['hello.py'])
```

📎 setup.py

Notice that this is ordinary Python. Let's go through it line by line...

1. When working with py2exe the only part of Distutils we'll typically need to reference directly is the setup function, so that's all we'll import.
2. Once Distutils is loaded, we need to load py2exe so that it can add its command.
3. Whitespace is good!
4. Call setup and tell it that we want a single console application and the main entry point is "hello.py".

3. Run your setup script

The next step is to run your setup script. Make sure to give the py2exe command and expect to see lots and lots of output:

```
C:\Tutorial>python setup.py py2exe
running py2exe
*** searching for required modules ***
*** parsing results ***
creating python loader for extension 'zlib'
creating python loader for extension 'unicodedata'
creating python loader for extension 'bz2'
```

```
*** finding dlls needed ***
*** create binaries ***
*** byte compile python files ***
byte-compiling C:\Tutorial\build\bdist.win32\winexe\temp\bz2.py to
bz2.pyc
byte-compiling C:\Tutorial\build\bdist.win32\winexe
\temp\unicodedata.py to unicodedata.pyc
byte-compiling C:\Tutorial\build\bdist.win32\winexe\temp\zlib.py to
zlib.pyc
skipping byte-compilation of c:\Python24\lib\StringIO.py to
StringIO.pyc

[skipping many lines for brevity]

skipping byte-compilation of c:\Python24\lib\warnings.py to
warnings.pyc
*** copy extensions ***
*** copy dlls ***
copying c:\Python24\lib\site-packages\py2exe\run.exe -> C:\Tutorial
\dist\hello.exe

*** binary dependencies ***
Your executable(s) also depend on these dlls which are not included,
you may or may not need to distribute them.

Make sure you have the license if you distribute any of them, and
make sure you don't distribute files belonging to the operating
system.

   ADVAPI32.dll - C:\WINDOWS\system32\ADVAPI32.dll
   USER32.dll - C:\WINDOWS\system32\USER32.dll
   SHELL32.dll - C:\WINDOWS\system32\SHELL32.dll
   KERNEL32.dll - C:\WINDOWS\system32\KERNEL32.dll

C:\Tutorial>
```

Two directories will be created when you run your setup script, build and dist. The build
directory is used as working space while your application is being packaged. It is safe to
delete the build directory after your setup script has finished running. The files in the dist
directory are the ones needed to run your application.

4. Test your executable

Now that the package has been created it is ready to test:

```
C:\Tutorial>cd dist

C:\Tutorial\dist>hello.exe
Hello World
```

Excellent, it works!!!

5. Providing the Microsoft Visual C runtime DLL

The Python interpreter was compiled using Microsoft Visual C, so your new program
needs the Microsoft Visual C runtime DLL to run. If you have installed appropriate
versions of Python or Visual Studio, then you will already have this DLL on your

computer. If some of your users might not already have this DLL, then they will not be able to run your program. The methods you may use to solve this depend on the version of Python you are using:

5.1. Python 2.4 or 2.5

If you are using Python 2.4 or 2.5, then the DLL you need is called MSVCR71.dll. This DLL will probably already have been included in your dist directory, in which case you need do nothing more.

However, the copyright on this file is owned by Microsoft, and you need to check whether you have the legal right to redistribute it. If you have a copy of Visual Studio, check the file redist.txt provided within the installation to see whether you have redistribution rights for this DLL. Generally you have the right to redistribute it if you own a license for Microsoft Visual C++, but not if you use the Express Editions.

If you do not have the rights to redistribute MSVCR71.dll, then your users must install it for themselves, using the 🌐 Microsoft Visual C++ 2005 Redistributable Package (vcredist_x86.exe).

Either you can instruct your users to download and run this themselves, or you could create an installer for your application (see Step 6 below), that includes vcredist_x86.exe (which is itself redistributable by anyone), and then run that as part of your application installation.

5.2. Python 2.6, 2.7, 3.0, 3.1

For Python 2.6, the DLL you need is called MSVCR90.dll. Py2exe is not able to automatically include this DLL in your dist directory, so you must provide it yourself.

To complicate things, there is more than one version of this DLL in existance, each with the same filename. You need the same version that the Python interpreter was compiled with, which is version 9.0.21022.8. Through the remainder of these instructions, hover your mouse over the dll file (or the vcredist_x86.exe installer executable) to confirm which version you've got. You'll need the vcredist_x86.exe that contains the 🌐 Microsoft Visual C++ 2008 Redistributable Package published 29-11-2007, so not the VS2008 SP1 one (tested with Python 2.7.1).

As for older versions of Python, you need to check redist.txt within your Visual Studio installation to see whether you have the legal right to redistribute this DLL. If you do have these rights, then you have the option to bundle the C runtime DLL with you application. If you don't have the rights, then you must have your users run the redistributable C runtime installer on their machines.

5.2.1. Bundling the C runtime DLL

If you do have the rights to redistribute MSVCR90.dll, there should be a copy of it in your Visual Studio install, under VC\redist\x86\Microsoft.VC90.CRT. Since Visual Studio 2008, you can't just copy this DLL file - you also need the manifest file that you'll find there. The redist.txt file states that you must distribute all three dlls and the unmodified manifest file and it is a violation of the license agreement to distribute only one of the dlls without the others (though py2exe only needs MSVCR90.dll.) The pertinent passage from

the redist.txt file is as follows:

> "For your convenience, we have provided the following folders for use when
> redistributing VC++ runtime files. Subject to the license terms for the software, you
> may redistribute the folder (unmodified) in the application local folder as a
> sub-folder with no change to the folder name. You may also redistribute all the files
> (*.dll and *.manifest) within a folder, listed below the folder for your convenience,
> as an entire set."

You must make py2exe copy the three dlls and the manifest file into your project's dist
directory, in a subdirectory called 'Microsoft.VC90.CRT'. To achieve this, add a data_files
option to your project's setup.py:

```
  from glob import glob
  data_files = [("Microsoft.VC90.CRT", glob(r'C:\Program
Files\Microsoft Visual Studio 9.0\VC\redist\x86\Microsoft.VC90.CRT
\*.*'))]
  setup(
    data_files=data_files,
    etc
  )
```

With this in place, running py2exe should put the files into your dist directory:

```
  dist
  |
  +-Microsoft.VC90.CRT
  | |
  | +-Microsoft.VC90.CRT.manifest
  | +-msvcm90.dll
  | +-msvcp90.dll
  | +-msvcr90.dll
  |
  |-etc
```

Now, simply copying the whole dist directory to your users machines should now allow
your application to run, even on machines that don't have their own copy of the C++
runtime.

Note that this method of including the C runtime is used by several Visual C++
applications - if you search your Program Files folder for msvcr90.dll, you may find
several applications that have this DLL and the associated manifest bundled alongside
their executable like this.

Also note that despite all the above, py2exe will complain that it cannot find
MSVCP90.dll. You must edit your setup.py to add the path to the dlls to the sys.path, e.g.

```
    sys.path.append("C:\\Program Files\\Microsoft Visual Studio
9.0\\VC\\redist\\x86\\Microsoft.VC90.CRT")
```

5.2.1.1 win32ui special case

win32ui needs MFC DLLs to run .exe, see Py2exeAndWin32ui for extra informations

5.2.2. Running the redistributable C runtime installer

If you don't have rights to redistribute MSVCR90.dll, then your users may install it on their machine by running the 🌐 Microsoft Visual C++ 2008 Redistributable Package (vcredist_x86.exe). It is important not to use the SP1 version of this installer, which contains the wrong version of MSVCR90.dll.

Either you can instruct your users to download and run this themselves, or you could create an installer for your application (see step 6 below), that includes vcredist_x86.exe (which is itself redistributable by anyone), and then run that as part of your application installation.

The installer puts a copy of the DLLs in the directory C:\WINDOWS\WinSxS (XP), inside subdirectories with mangled names. The manifest file is in the 'Manifests' subdirectory, again this will have a mangled filename. You can still discern the text 'Microsoft.VC90.CRT' and '9.0.21022.8' within the mangled file and directory names, to find the files. It is possible to take a copy of these files and remove the filename mangling, to embed them in your application as described in 5.2.1.

6. Build an installer if applicable

py2exe is not an installer builder - it merely assembles the files needed to run your Python program. There are plenty of good installer builders out there including some that are open source (e.g., 🌐 NSIS) and some that are free (e.g., 🌐 Inno Setup).

Tutorial (última edición 2011-03-14 18:21:19 efectuada por MikeFox)