How to create a Windows Service in Python

Davide MastromatteoAug 1, 2018

Hi guys, today's post is just for the ones of you that work with the "OS of the misoriented slashes": Microsoft Windows. :)

Have you ever had the need of writing a Python script that could run in background as a Windows Service? In this post you will learn how to do it in less than 10 minutes, no jokes.

I will skip all the introduction about Windows Services, how convenient they could be, how much could be appreciated the fact that they can be run in background even when the user is logged off etc... I mean, if you can code in Python and you use Windows I bet you already know what a Windows Service is, don't you?

So, first of all, let's start by installing the Python for Windows extensions:

```
pip install pywin32
```

Once you have done it, let's write this base class, your Windows service will be a subclass of this base class.

```
'''
   SMWinservice
   by Davide Mastromatteo
   Base class to create winservice in Python
   ----------------------------------------
   Instructions:
   1. Just create a new class that inherits from this base class
   2. Define into the new class the variables
      _svc_name_ = "nameOfWinservice"
      _svc_display_name_ = "name of the Winservice that will be displayed in scm"
      _svc_description_ = "description of the Winservice that will be displayed in scm"
   3. Override the three main methods:
       def start(self) : if you need to do something at the service initialization.
                         A good idea is to put here the inizialization of the running
   condition
       def stop(self)  : if you need to do something just before the service is stopped.
                         A good idea is to put here the invalidation of the running
   condition
       def main(self)  : your actual run loop. Just create a loop based on your running
   condition
   4. Define the entry point of your module calling the method "parse_command_line" of the
   new class
   5. Enjoy
   '''


   import socket


   import win32serviceutil


   import servicemanager
   import win32event
   import win32service




   class SMWinservice(win32serviceutil.ServiceFramework):
       '''Base class to create winservice in Python'''


       _svc_name_ = 'pythonService'
       _svc_display_name_ = 'Python Service'
       _svc_description_ = 'Python Service Description'


       @classmethod
       def parse_command_line(cls):
           '''
           ClassMethod to parse the command line
           '''
           win32serviceutil.HandleCommandLine(cls)


       def __init__(self, args):
           '''
           Constructor of the winservice
           '''
           win32serviceutil.ServiceFramework.__init__(self, args)
           self.hWaitStop = win32event.CreateEvent(None, 0, 0, None)
           socket.setdefaulttimeout(60)
```

```
    def SvcStop(self):
        '''
        Called when the service is asked to stop
        '''
        self.stop()
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        win32event.SetEvent(self.hWaitStop)


    def SvcDoRun(self):
        '''
        Called when the service is asked to start
        '''
        self.start()
        servicemanager.LogMsg(servicemanager.EVENTLOG_INFORMATION_TYPE,
                              servicemanager.PYS_SERVICE_STARTED,
                              (self._svc_name_, ''))
        self.main()


    def start(self):
        '''
        Override to add logic before the start
        eg. running condition
        '''
        Pass


    def stop(self):
        '''
        Override to add logic before the stop
        eg. invalidating running condition
        '''
        Pass


    def main(self):
        '''
        Main class to be ovverridden to add logic
        '''
        Pass


# entry point of the module: copy and paste into the new module
# ensuring you are calling the "parse_command_line" of the new created class
if __name__ == '__main__':
    SMWinservice.parse_command_line()
```

Let's examine the class we have just introduced a little.

- ***def SvcDoRun(self)***: it's the method that will be called when the service is requested to start.
- ***def SvcStop(self)***: it's the method that will be called when the service is requested to stop.
- ***def start(self):*** it's a method that you will be asked to override if you need to do something when the service is starting (before started)
- ***def stop(self)***: it's the method that you will be asked to override if you need to do something when the service is stopping (before stopped)
- ***def main(self)***: it's the method that will contain the logic of your script, usually in a loop that keeps it alive until the service is stopped.
- ***def parse_command_line(cls)***: it's the method that handle the command line interface that you can use to install and update your windows service

Can you see how easy it is with pywin32 to interface with the system to create a Windows Service? The last mention is for the following variables:

```
_svc_name_ = "PythonCornerExample"
_svc_display_name_ = "Python Corner's Winservice Example"
_svc_description_ = "That's a great winservice! :)"
```

These are just three variables that contain the name of the service, the "friendly name" that will be used by Windows to display the name on the mmc console and a short description of your service.
As always, enough talk, let's code something useful!

Let's pretend that we want to create a Winservice that, when started, creates a random file on our C:\ drive each 5 seconds.

What? Do you think it is stupid? Well, install it on your boss PC, set the destination folder as its user's desktop and you will change your mind. :)

However, how can you achieve this result? Super easy.

Step 1: Subclass the SMWinservice class we have just met.
Step 2: On the new class, override the three variables *_svc_name_*, *_svc_display_name_* and *_svc_description_*.
Step 3: Override the "***start***" method to set the running condition. Setting a boolean variable will be enough for that.
Step 4: Override the "***stop***" method to invalidate the running condition when the service is requested to be stopped.
Step 5: Override the "***main***" method to add the logic of creating a random file every 5 seconds
Step 6: Add the call at the "***parse_command_line***" function to handle the command line interface.

The result should be something like this:

```
import random
import time
from pathlib import Path
from SMWinservice import SMWinservice


class PythonCornerExample(SMWinservice):
    _svc_name_ = "PythonCornerExample"
    _svc_display_name_ = "Python Corner's Winservice Example"
    _svc_description_ = "That's a great winservice! :)"


    def start(self):
        self.isrunning = True


    def stop(self):
        self.isrunning = False


    def main(self):
        i = 0
        while self.isrunning:
            random.seed()
            x = random.randint(1, 1000000)
            Path(f'c:\\{x}.txt').touch()
            time.sleep(5)


if __name__ == '__main__':
    PythonCornerExample.parse_command_line()
```

That's it! Now it's time to install our newly created winservice. Just open a command prompt, navigate to your script directory and install the service with the command:

```
> python PythonCornerExample.py install
Installing service PythonCornerExample
Service installed
```

In future, if you want to change the code of your service, just modify it and reinstall the service with

```
> python PythonCornerExample.py update
Changing service configuration
Service updated
```

Now, open the "Services" msc snap in

```
> mmc Services.msc
```

locate your new *PythonCornerExample* winservice, and right click and choose properties.
Here you can start your service and configure it at your will.
Now try to start your service and go to see your C:\ folder contents.
Can you see al these files being created to your C:\ folder? Yeah, that's working!
But now it's time to stop it! :)
You can do it from the previous windows or just by using the command line

```
> net stop PythonCornerExample
Il servizio Python Corner's Winservice Example sta per essere
arrestato..
Servizio Python Corner's Winservice Example arrestato.
```

If somethings goes wrong…

There are a couple of known problems that can happen writing Windows Services in Python. If you have succesfully installed the service but starting it you get an error, follow this iter to troubleshoot your service:

1. Check if Python is in your PATH variable. It MUST be there. To check this, just open a command prompt and try starting the python interpreter by typing "python". If it starts, you are ok.

2. Be sure to have the file **C:\Program Files\Python36\Lib\site-packages\win32\pywintypes36.dll** (please note that "36" is the version of your Python installation). If you don't have this file, take it from **C:\Program Files\Python36\Lib\site-packages\pywin32_system32\pywintypes36.dll** and copy it into **C:\Program Files\Python36\Lib\site-packages\win32**

3. If you still have problems, try executing your Python script in debug mode. To try this with our previous example, open a terminal, navigate to the directory where the script resides and type:

```
> python PythonCornerExample.py debug
```

Ok, that's all for today, this was just a brief introduction to developing Windows Services with Python. Try it by yourself and … happy coding!
D.

Source

https://medium.com/the-python-corner/how-to-create-a-windows-service-in-python-88ca534ce76b