Creating a Windows service using Python

All Versions

Headless processes (with no UI) in Windows are called Services. They can be controlled (started, stopped, etc) using standard Windows controls such as the command console, Powershell or the Services tab in Task Manager. A good example might be an application that provides network services, such as a web application, or maybe a backup application that performs various background archival tasks. There are several ways to create and install a Python application as a Service in Windows.

Examples

A Python script that can be run as a service

The modules used in this example are part of pywin32 (Python for Windows extensions). Depending on how you installed Python, you might need to install this separately.

```
import win32serviceutil
import win32service
import win32event
import servicemanager
import socket


class AppServerSvc (win32serviceutil.ServiceFramework):
    _svc_name_ = "TestService"
    _svc_display_name_ = "Test Service"

    def __init__(self,args):
        win32serviceutil.ServiceFramework.__init__(self,args)
        self.hWaitStop = win32event.CreateEvent(None,0,0,None)
        socket.setdefaulttimeout(60)

    def SvcStop(self):
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        win32event.SetEvent(self.hWaitStop)

    def SvcDoRun(self):
        servicemanager.LogMsg(servicemanager.EVENTLOG_INFORMATION_TYPE,
                              servicemanager.PYS_SERVICE_STARTED,
                              (self._svc_name_,''))
        self.main()

    def main(self):
        pass

if __name__ == '__main__':
    win32serviceutil.HandleCommandLine(AppServerSvc)
```

This is just boilerplate. Your application code, probably invoking a separate script, would go in the main() function.

You will also need to install this as a service. The best solution for this at the moment appears to be to use Non-sucking Service Manager . This allows you to install a service and provides a GUI for configuring the command line the service executes. For Python you can do this, which creates the service in one go:

```
nssm install MyServiceName c:\python27\python.exe c:\temp\myscript.py
```

Where my_script.py is the boilerplate script above, modified to invoke your application script or code in the main() function. Note that the service doesn't run the Python script directly, it runs the Python interpreter and passes it the main script on the command line.

Alternatively you can use tools provided in the Windows Server Resource Kit for your operating system version so create the service.

Running a Flask web application as a service

This is a variation on the generic example. You just need to import your app script and invoke it's `run()` method in the service's `main()` function. In this case we're also using the multiprocessing module due to an issue accessing `WSGIRequestHandler` .

```
import win32serviceutil
import win32service
import win32event
import servicemanager
from multiprocessing import Process

from app import app
```

```
class Service(win32serviceutil.ServiceFramework):
    _svc_name_ = "TestService"
    _svc_display_name_ = "Test Service"
    _svc_description_ = "Tests Python service framework by receiving and echoing messages over a

    def __init__(self, *args):
        super().__init__(*args)

    def SvcStop(self):
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        self.process.terminate()
        self.ReportServiceStatus(win32service.SERVICE_STOPPED)

    def SvcDoRun(self):
        self.process = Process(target=self.main)
        self.process.start()
        self.process.run()

    def main(self):
        app.run()


if __name__ == '__main__':
    win32serviceutil.HandleCommandLine(Service)
```

Adapted from http://stackoverflow.com/a/25130524/318488

Syntax

Parameters

Remarks