

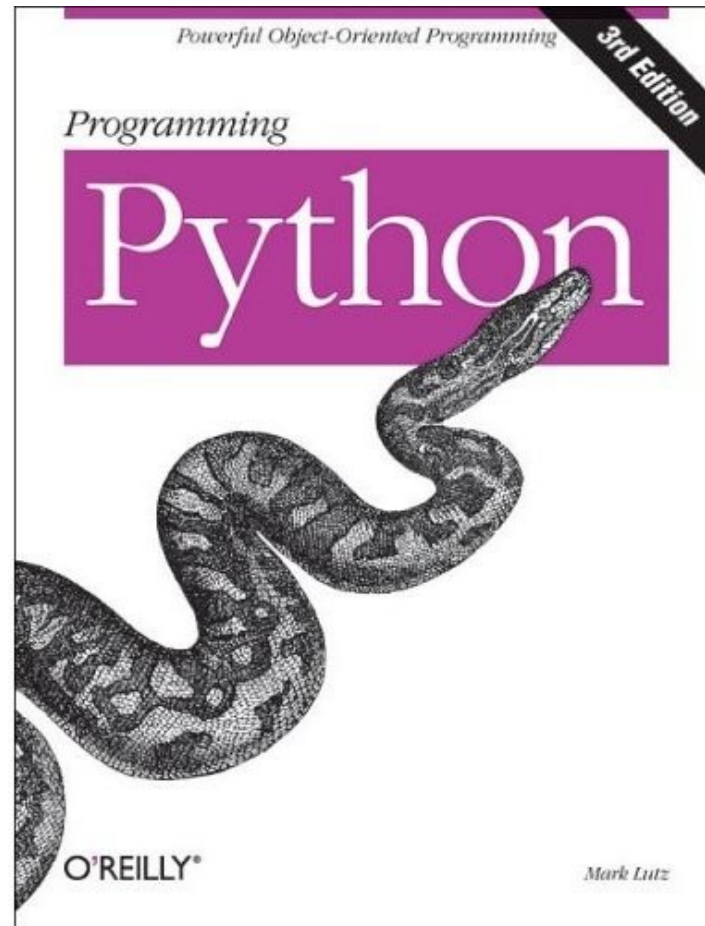
Twistedのお話

Twistedって何？

おおたに(ありえるねっとわーく)

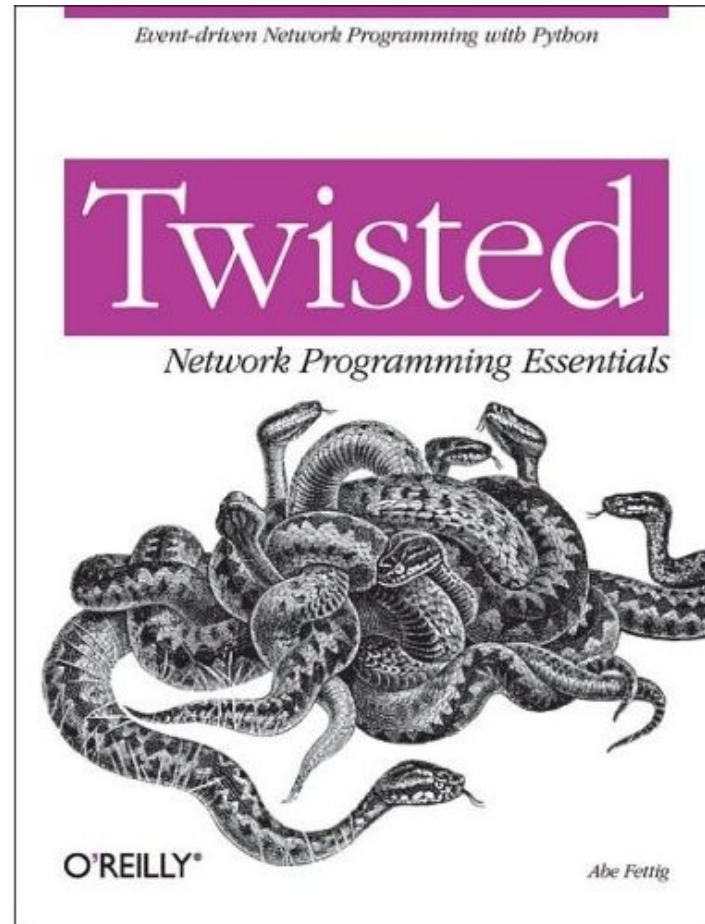
その前にPythonって？

Pythonって言うのはこれ



それじゃTwistedって？

Twistedって言うのはこれ



つまりは、

グロテスク?

Twistedと言うのは?

イベント駆動型ネットワークプログラミングの
フレームワーク・ライブラリ

どんなものを使っているの

- BitTorrent
- AppleのiCal WebDAVサーバ
- Xen
- Zope?
- 他たくさん

ネットワークプログラミングの スタイル

- 同期型
 - ブロッキングIO
- 非同期型
 - ノンブロッキングIO
 - select/poll
 - イベント駆動

同期型って？

```
from httplib import HTTPConnection
```

```
>>> conn = HTTPConnection("www.liris.org")
```

```
>>> conn.request("GET", "/index.html")
```

```
>>> r1 = conn.getresponse()
```

```
>>> print r1.status, r1.reason
```

```
200 OK
```

```
>>> data1 = r1.read()
```

非同期型って？

- 長くなるので例はあとで
- Connectをコールしたときに待たない
- Connectが成功したときにイベントがあがる
- ネットワークが読み書きできるようになるとイベントがあがる(正確にはちょっと違うけど)

なんで二つあるの？

- いろいろな理由
- 同期型はプログラミングが簡単
- 非同期型はとってもめんどくさい

めんどくさいものを使う理由

- ネットワークとCPU効率の問題
 - ネットワークプログラミングはとにかく待ちが多い
 - スレッド・プロセスのコンテキストの切替えはコストがかかる
- 同期型で効率を考えると → スレッドを沢山作る
- 非同期型だと一つのスレッドで複数の接続を処理できる(一つのスレッドで全てできる)
- 効率の問題はいろいろな論文やベンチマークの結果があるけど、real worldで・・・

僕が非同期でやりたい理由

- マルチスレッドでの処理は、スレッド間の同期が大変。人のやるプログラミングじゃない！
 - スレッドの切り替わるタイミングなどがlinux/Windowsで違う。Macは知らない。
 - デバッグできない。デバッグ(観測)すると動作が変わる。
 - 根気がないとちゃんとしたものはできない
- スレッド間で協調しない場合は、マルチスレッドもあり。

DNSの逆引き

```
import threading
import Queue
import socket
import sys
import fileinput

def lookup(ip_address):
    try:
        return socket.gethostbyaddr(ip_address)[0]
    except socket.herror:
        return ip_address

def resolver(queue, lock):
    while True:
        ip_address = queue.get()
        if ip_address == None:
            break
        host_name = lookup(ip_address)
        lock.acquire()
        try:
            print host_name
        finally:
            lock.release()
```

```
def main():
    queue = Queue.Queue()
    num_threads = int(sys.argv[1])
    for line in fileinput.input(sys.argv[2]):
        ip_address = line.strip()
        queue.put(ip_address)
    lock = threading.Lock()
    for i in xrange(num_threads):
        queue.put(None)
        thread = threading.Thread(target = resolver, args = (queue,
            lock))
        thread.start()
```

main()

出展 : いやなブログ

<http://0xcc.net/blog/archives/000099.html>

Twisted版DNSの逆引き

```
from twisted.names import client
from twisted.python import util, log
from twisted.internet import reactor,defer
import fileinput
def handleResult(results):
    for result in results:
        success, r = result
        if success:
            print r[0][0].payload.name
    reactor.stop()
if __name__ == "__main__":
    import sys
    dlist = []
    for addr in fileinput.input(sys.argv[1]):
        addr = addr.strip()
        ptr = ''.join(addr.split('.')[:-1]) + '.in-addr.arpa'
        c = client.lookupPointer(ptr)
        dlist.append(c)
    defer.DeferredList(dlist,
        consumeErrors=True).addCallback(
        handleResult)
    reactor.run()
```

- 最大待ち時間は一番名前解決に一番遅いもの
- 待ち時間に別の処理を行うことも可能

WebChat + AJAXの場合

- 同期型

- AJAXで定期的にPolling

- ネットワーク負荷

- 即時性

- AJAXで送信専用接続と受信専用接続(新たなデータが来た地点でレスポンスを返す)

- 接続され続けるのでクライアント数に応じたスレッドが必要

WebChat + AJAXの場合

- 非同期型

- AJAXで送信専用接続と受信専用接続(新たなデータが来た地点でレスポンスを返す)
- 問題は？うーん、あんまりないかも
- 普通のWebサーバじゃできないんだよね
- つまりは、とっても優れている

非同期の問題

- すべて一つのスレッドで行う
 - 重い処理があるとプログラム全体をブロックする
 - Zopeを完全にシングルスレッドで動かすとどうなるかな?
 - 重い処理はスレッドを別に作らないと効率が却って落ちる
 - スレッド間の同期の問題が発生して大変

なので

- どちらが優れているかは一概にはいえない
- どちらのプログラミングモデルを使うかは、作りたいものによる
- 魔法の杖は今のところない

非同期プログラミングと言うと

- Twisted
- 標準ライブラリのasyncoreが有名

Pythonの標準ライブラリじゃダメなの？

- つまりは、`asyncore`など。
- 一つの処理(HTTPサーバだけとか)なら十分
- 複数の接続 + 複数のプロトコルを扱うと力不足。
- `asyncore`はAPIが嫌い

Twistedは？

- ここから本題
- 複数の接続、複数のプロトコルを統一的に扱える
- 低レベルなことから高レベルなAPIまで用意されていて、好みに応じて使い分けられる
- 低レベルなAPIが好き

Twistedの恐ろしい所

- ネットワークはTCP/UDP/Unix Socketなどをサポート
- ファイルアクセスも非同期処理できる
- データベースも非同期処理できる！
- Linuxならコンソール(標準入出力)まで非同期処理できる！
 - 標準出力を非同期でやるとめんどくさいだけ！
 - 大量に出力しない限り誤差の範囲
- 待ちが発生しそうなことは大抵非同期でできる
- select/pollを使わずにWin32のイベント処理に置き換えられる

Twistedの基本

- Reactor
- Deferred
- Factory
- Protocol

Reactorとは？

- イベントループ (Win32 event loopのようなもの)
- イベントに応じてコールバックを実行
- スケジュール管理
- select/pollだけじゃなく、Win32, GTK1,2, QTなどのevent loopなども使用可能 → クライアントのGUIプログラミングと親和性がよい

Deferredって？

- コールバック関数、エラーバック関数を複数登録
- 登録したものが順次実行される

```
import twisted.web.client
def f1(data):
    print data
def f2(data):
    pass
def f3(error):
    print str(error)
from twisted.internet import reactor
deferred = twisted.web.client.getPage("http://www.liris.org")
deferred.addCallback(f1).addCallback(f2).addErrback(f3)

reactor.run()
```

factoryとprotocol

- Factoryは接続が確立されたときにprotocolオブジェクトを作る
- Protocolオブジェクトは接続中は生存。接続が終わると破棄。
- protocolオブジェクトは、ネットワークに送受信されるデータのハンドリングを行う。
- Protocolオブジェクト間のデータの受け渡しはfactoryを介して行うらしい。

Echoサーバなら

```
from twisted.internet.protocol import  
    Protocol
```

```
class Echo(Protocol):
```

```
    def connectionLost(self, reason):  
        self.factory.numProtocols =  
        self.factory.numProtocols-1
```

```
    def dataReceived(self, data):  
        self.transport.write(data)
```

```
from twisted.internet.protocol import  
    Factory  
from twisted.internet import reactor
```

```
factory = Factory()  
factory.protocol = Echo
```

```
reactor.listenTCP(1234, factory)
```

```
reactor.run()
```

高レベルなこと

- 自分でHTTPやSMTP,IRCのProtocolを自分で書くのは面倒
- Twistedのサブプロジェクトが多くのプロトコルをサポートしている(ものによってできは様々)
- Web関係はHTTP Protocolの上にさらに独自のフレームワークを用意

さらに高レベルなこと

- 独自のスクリプトの仕組みがあるらしい
- アプリケーションのフレームワークがあるらしい
- Unix DaemonやWindows Serviceに簡単にできる

おまけ

- Twistedはサーバで使える
- Twistedはクライアントでも使える
- クライアントではTwistedとpy2exeとかを使えば配布が楽。