wwty

wwty.iteye.com/blog/400971

一、Twisted基本模型
Twisted 网络编程框架是一种基于事件的网络编程框架，用户需要继承特定的类，并重载其中的方法来处理网络通信

中可能出现的各种情况。

Twisted的网络通信模型 最基本的也要由三部分组成：**反应器(reactor)、协议(protocol)、工厂(factory)。**

1、其中反应器reactor用来执行事件循环，分发事件处理等 等，每个应用程序中一般只能启动一个reactor。
2、协议用来完成与一个已经连接成功的主机的交互功能，主要有数据的接收和发送功能。连接的断开事件也可以在这

里处理。注意是在连接建立之后protacal才开始负责下面的工作的。
3、工厂负责与一个协议的启动和关闭功能，而且还负责在连接成功时生成一个协议对象，(by gashero)用于与远程主

机的交互功能。

一个典型的Twisted应用程序会建立至少一个协议，可以从twisted.internet.protocol.BaseProtocol类或其子类继承

。协议还需要实现数据的接收处理，即收到数据之后需要做出何种响应。比较简单的Twisted应用程序可以继承一个空

的工厂，来自 twisted.internet.protocol.Factory或其子类。工厂至少应该指定protocol属性，指向协议类。最后

就是要启动事件 循环，根据连接方向的不同，可以选择用reactor的connectXXX()或listenXXX()方法，然后执行

reactor.run()启动 事件循环。

二、协议模型
所有协议类的基类是 twisted.internet.protocol.BaseProtocol，但是一般使用其子类。不同的协议子类提供了不同

的数据接收方法，如 LineReceiver子类就允许同时使用行和原始数据两种方法接收数据，使用非常方便。

BaseProtocol的接口如下：
class BaseProtocol:
connected=0 #是否已经连接了
transport=None #用于数据发送的传输对象
def makeConnection(self,transport): #建立连接的方法，不是事件方法，一般不要重载
def connectionMade(self): #连接成功事件，可重载
可以看到BaseProtocol可以理解为一个虚基类，实现的功能十分简陋。实际的应用程序一般也不是直接继承
BaseProtocol来实现协议，而是继承Protocol类。Protocol类提供了基本完善的协议功能，接口定义如下：

class Protocol(BaseProtocol):
def dataReceived(self,data): #接收到数据事件，可重载
def connectionLost(self,reason=connectionDone): #连接断开事件，可重载，依靠reason区分断开类型

从Protocol类继承就可以完成协议的基本处理了，包括连接的建立和断开事件，还有数据接收事件。

三、工厂模型
相对于协议，工厂可以发挥的空间就很小了。所有工厂的基类是twisted.internet.protocol.Factory。这个类定义了三个方法，接口如下：
class Factory:
protocol=None #指向一个协议类
def startFactory(self): #开启工厂
def stopFactory(self): #关闭工厂
def buildProtocol(self,addr): #构造协议对象，并给协议对象添加一个factory属性指向工厂，可以重载
从 这里可以看到，工厂类中最重要的部分就是protocol属性，将这个属性设置为一个协议类(注意不是协议对象)，就

可以将这个工厂设置为对应协议的工厂 了。前两个方法控制工厂的开启和关闭，用于资源的初始化和释放，可以重载

。buildProtocol()方法可以控制协议对象的生成，(by gashero)如果需要多传递一个属性，可以重载，但是重载时应

该注意在方法内继承原方法内容。

工厂还分为客户端工厂和服务器工厂。服务器工厂继承自Factory，而没有任何修改，定义如下：
class ServerFactory(Factory):

客户端工厂则有较多内容，接口定义如下：
class ClientFactory(Factory):
def startedConnecting(self,connector): #连接建立成功时
def clientConnectionFailed(self,connector,reason): #客户端连接失败
def clientConnectionLost(self,connector,reason): #连接断开
这三个方法都传递了一个connector对象，这个对象有如下方法可用：
connector.stopConnection() #关闭会话
connector.connect() #一般在连接失败时用于重新连接

四、连接器
连接器指客户端用来连接的包装。
twisted.internet.protocol.ClientCreator是一个连接器，用来连接远程主机，接口定义如下：
class ClientCreator:
def __init__(self,reactor,protocolClass,*args,**kwargs):
def connectTCP(self,host,port,timeout=30,bindAddress=None):
def connectUNIX(self,address,timeout=30,checkPID=0):
def connectSSL(self,host,port,contextFactory,timeout=30,bindAddress=None):
三个连接方法都是返回Deferred对象作为Protocol实例，在不需要工厂时可以直接使用这个类来产生仅使用一次的客

户端连接。这时，协议对象之间没有共享状态，也不需要重新连接。

在Twisted中，有一种特殊的对象用于实现事件循环。这个对象叫做reactor。
可以把反应器(reactor)想象为Twisted程序的中枢神经。
reactor根据平台的不同，提供了不同的实现，在使用的时候，可以根据平台的不同，选择不同的reactor
除了分发事件循环之外，反应器还做很多重要的工作：定时任务、线程、建立网络连接、监听连接。
为了让反应器可以正常工作，需要启动事件循环。

```
from twisted.internet import reactor
print 'Running the reactor ...'
reactor.run()
print 'Reactor stopped.'
```

这样就建立了一个事件循环。

reactor.callLater方法用于设置定时事件：
reactor.callLater函数包含两个必须参数，等待的秒数，和需要调用的函数
意思是多少秒钟之后调用某个函数
在实际应用中，reactor.callLater是常用于超时处理和定时事件。可以设置函数按照指定的时间间隔来执行关闭非活

动连接或者保存内存数据到硬盘。

reactor.stop()停止循环，退出循环