www.photocase.com

Creating network apps with Twisted

TWISTING PYTHON

The Twisted framework makes it so easy to create network-aware applications in Python. Twisted speaks all the major Internet protocols, from mail through chat, and it can handle encryption. We'll show you how to set up a personal web server with Twisted. **BY MARKUS FRANZ**

Most programmers eventually face the task of adding network communication capabilities to their applications. If it is simply a case of manipulating web content, there are simple answers, but more complex functions, such as adding full-fledged email capabilities or a complete web server, involve much more effort.

Although the Python standard library has modules to match most walks of a programmer's daily life (batteries included, in Python-speak), special applications require external packages. Twisted [1] is a well-organized and powerful collection of modules for adding networking capabilities to Python programs.

If you are coding an email client with multiple protocol support (POP 3, SMTP, IMAP), the Twisted framework will remove the need to start from scratch. Twisted Mail has all the major mail protocols. Twisted also has ready-to-run modules for SSH, SFTP, HTTP (including HTTP/1.1), DNS, NNTP, and Jabber. If you insist on re-inventing the wheel – by implementing your own protocols for example – the *twisted.cred* and *twisted.spread* Twisted modules can help to simplify the job.

Asynchronous Networking

Twisted is basically an asynchronous network framework. In contrast to other libraries, the Twisted functions do not block when they are called. Applications just keep on running until they are told that the required data is available. Although Python's standard libraries could handle this (the *asyncore* module has basic functionality for switching between multiple I/O channels within a thread), Twisted implements this design at a higher level in its protocols, interfaces, and components. This lets programmers write network applications that do without additional processes and threads while at the same time handling multiple I/O channels.

Highly Modular

To use Twisted to make your own software network-aware, you first need to find out which part of the framework you will need. The developers split Twisted into multiple sub-projects when moving from version 1 to 2, with the aim of adding more clarity. Table 1 has a list of the most important elements, and there is a complete list at [2].

Twisted requires the Zope Interface module, which implements interfaces

Table 1: Major Twisted Modules

twisted	Framework for asynchronous applications, the basis for all Twisted subprojects
twisted.conch	Implementation of the SFTP and SSH protocols for clients and servers
twisted.web	HTTP protocol for clients and servers
twisted.web2	Support for the HTTP/1.1 protocol as a server framework; this package is still under development at present, and should not be used for critical applications
twisted.mail	Implementation of the SMTP, IMAP, and POP protocols for clients and servers
twisted.names	DNS protocol support for clients and servers
twisted.news	NNTP protocol for clients and servers
twisted.words	Module for chat or instant messaging applications

that the Python language core lacks. If you just need to use a single project, such as Twisted Web or Twisted Mail, you can download the required module from the project homepage. Alternatively, Twisted Sumo [3] has all (stable) modules, including a Zope interface.

After unpacking, give the *python setup.py install* command. If the Zope interface is not installed, the installer will display an error message and quit.

The *twisted.cred* module handles authentication in client-server communications. It allows multiple network protocols to connect to a system, to authenticate, and to exchange data. For example, POP 3 support in Twisted provides a combination of the username and password to open the requested mailbox. The so-called Perspective Broker is important here; the broker provides access to remote objects and implements object copying, referencing, and caching.

Database Connection

twisted.enterprise provides a database interface that is compatible with Python-DB-API 2.0. This makes access to MySQL, Oracle, or PostgreSQL databases child's play. The module uses an asynchronous interface, which can run in multiple threads, without losing thread safety in an event-based Twisted main loop. The main loop occurs within the *twisted.internet* module and is known as the *reactor*. It implements the infinite loop of the program in which Twisted handles various events. The reactor provides the underlying interface to Twisted's major internal capabilities, such as network connections, threading, or event handling.

Other modules, such as *twisted.protocols* or *twisted.manhole*, are very rarely needed in practical applications. Conch implements version 2 of the Secure Shell protocol for Twisted. The how-to at [5] discusses the implementation of an SSH client with Conch in a few simple steps.

Web or application servers are one of Twisted's most interesting fields of application and use *twisted.web* or *twisted.web2*. The powerful template toolkit for this task is titled Nevow [6]. Twisted

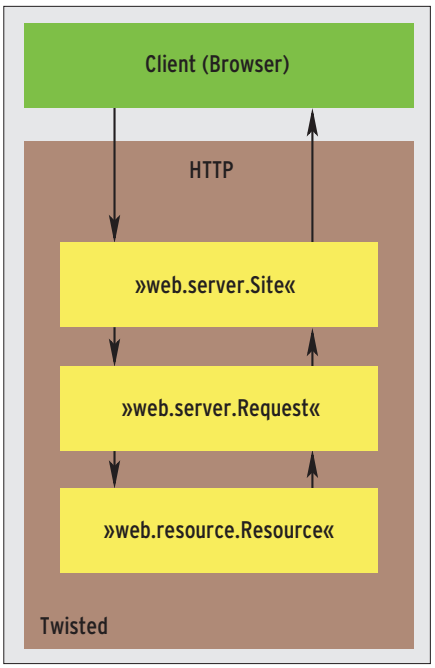

Figure 1: Web request process within the Twisted framework.

also has the full range of functions required for programming HTTP clients. The *twisted.web* API supports multiple

abstraction layers: from simple web servers, through session support, interacting application servers, and distributed websites – the options are many.

When a client request reaches a web server, the server creates a request object and hands the object back to the resource system, which creates the response (Figure 1).

Besides *twisted.web*, there is also the *twisted.web2* module, which is still under development. The major improvements are as follows:

- HTTP 1.1 support
- Internal and predefined output filters, for gzip-compressed serving of web pages
- Separation of high and low level request handling
- Correct analysis of HTTP headers
- Vastly improved URL rewriting, if used in combination with a proxy

Although there are major enhancements in *twisted.web2* in comparison to the previous version, the developers do not recommend using the new module at present, one of the reasons being that the new module is slower than its predecessor.

The last major module is Twisted Mail, which implements SMTP, POP 3, and IMAP 4. Besides protocols, the module can also read and write the Maildir mailbox format. There is also a preconfigured combination of SMTP and POP 3 for mail servers and virtual hosting systems. And Twisted Mail understands most Sendmail options, which can come in handy for downwardly compatible applications.

Listing 1: Simple Web Server

```
01 # Load modules
02 from twisted.internet import
   reactor
03 from twisted.web import
   static, server
04
05 # Set root directory
06 my_server = static.File('/var/
   www/htdocs')
07
08 # Launch web server on port
   7777
09 reactor.listenTCP(7777,
   server.Site(my_server))
10 reactor.run()
```

Frozen Server

The Twisted framework has a collection of command line programs, which prepare Twisted applications for special scenarios [7]. The *mktap* and *tapconvert* tools create Tap, Tas, or Tax formatted files from the Python source code. The code can then be used with the various Twisted servers, for example: Web, FTP, or IRC.

The *twistd* tool brings Tap files to life. Strictly speaking, *twistd* is not required to run Twisted applications, but it does make things easier, as it takes control of the reactor and handles starting and quitting the application. Additionally, *twistd* supports the selection of a different reactor type, allowing an application to run in daemon mode or write logfiles. The *tap2deb* and *tap2rpm* programs compress finished server applications for distribution in Debian or RPM package formats. The tools automatically generate scripts for installing and removing the server.

Listing 1 shows a simple web server and demonstrates how powerful Twisted is. You could use this code to enable web-based configuration of your application. The first couple of lines load the required modules. Line 6 creates a new server with a web root directory for HTML documents – */var/www/htdocs* in our example. The following instruction tells the reactor to listen for requests on port 7777. The *reactor.run()* call launches the web server.

Listing 2 demonstrates a web server that adds a number of options to the basic framework in Listing 1. First of all, lines 9 through 11 enable Perl script support: we want the server to hand files with a *.pl* extension to the Perl interpreter, */usr/bin/perl*, and to return its output. Support for PHP or other languages could just as easily be enabled at this point.

The *putChild()* method in line 14 sets the CGI directory to */var/www/cgi-bin*,

Listing 2: Extended Web Server

```
01 # Load modules
02 from twisted.internet import reactor
03 from twisted.web import static, server, twcgi
04
05 # set root directory
06 my_server = static.File('/var/www/htdocs')
07
08 # Evaluate Perl scripts
09 class PerlScript(twcgi.FilteredScript):
10         filter = '/usr/bin/perl' # path to Perl interpreter
11 my_server.processors = {'.pl': PerlScript}
12
13 # Set and enable CGI directory
14 my_server.putChild('cgis', twcgi.CGIDirectory('/var/www/cgi-bin'))
15
16 # Directories for other targets
17 my_server.putChild('doc', static.File('/var/www/doc'))
18
19 # Index files
20 my_server.indexNames = ['index.html', 'index.htm', 'index.pl']
21
22 # Launch web server on port 7777
23 reactor.listenTCP(7777, server.Site(my_server))
24 reactor.run()
```

thus allowing access to the scripts at *http://servername:7777/cgis*. The method also specifies the path for the *doc* directory, */var/www/doc*. The *indexNames* variable specifies which files the server looks for during a directory request. The order is defined by the filenames specified.

The following Twisted tools could be used as an alternative to the web server code:

```
mktap web ➥
--path /var/www/htdocs ➥
--port 7777
twistd --file web.tap
```

mktap creates a pre-configured Tap file. In this case, *mktap* sets the root directory to */var/www/htdocs* and the port to *7777*. The results end up in the *web.tap* file, which the *twistd* server program uses as a parameter. After launching, the server, the logfile, *twistd.log*, and the PID file, which can be used to kill the server (*kill `cat twistd.pid`*), reside in the current directory. *mktap web -help* gives you more information on the available options.

Simple but Powerful

This article can only give you a quick overview of Twisted. The framework, along with Twisted Mail, Conch, Twisted Web(2), and the other sub-projects, form the basis for many professional network applications – even NASA uses Twisted Matrix [8].

Helping Python applications reach for the stars makes life a lot easier for overworked programmers back here on earth. Twisted clearly reduces the effort involved in developing a client or server by removing the need to implement known protocols. This gives programmers secure, fast, stable, and flexible network applications at the click of a mouse. ■

THE AUTHOR

Markus Franz runs a business advisory service for companies involved in IT projects. He develops Internet search engines, including Metager 2, www.metager2.de, which was implemented in Python throughout. Markus is just 17 years old and still attends the Armin Knab High School at Kitzingen, Germany.

INFO

[1] Twisted Matrix project page: *http://www.twistedmatrix.com*

[2] Overview of all Twisted projects: *http://www.twistedmatrix.com/projects*

[3] Twisted Sumo: *http://twistedmatrix.com/projects/core*

[4] Zope Interface: *http://www.zope.org/Wikis/Interfaces*

[5] SSH client with Conch: *http://twistedmatrix.com/projects/conch/documentation/howto/conch_client.html*

[6] Nevow (template system): *http://divmod.org/projects/nevow*

[7] Tools: *http://twistedmatrix.com/projects/core/documentation/howto/basics.html*

[8] Twisted users: *http://twistedmatrix.com/services/success*