



Crear aplicaciones en red con Twisted

SERPIENTES

El entorno de trabajo Twisted facilita enormemente la creación de aplicaciones en red para Python. Twisted soporta los principales protocolos, desde email hasta chat, y admite encriptación. Os mostramos cómo configurar un servidor para páginas Web personales con Twisted. **POR MARKUS FRANZ**

La mayoría de los programadores se encuentran tarde o temprano con la tarea de añadir funcionalidades de comunicación a sus aplicaciones. Si se trata simplemente de modificar el contenido de una página Web, existeN soluciones sencillas. Pero funcionalidades más complejas, como añadir capacidad de correo electrónico o un servidor Web completo, requieren un esfuerzo mucho mayor.

Aunque la librería estándar de Python tiene módulos para la mayoría de los requerimientos de la vida diaria de un pro-

gramador (baterías incluidas, según la jerga Python), algunas aplicaciones especiales pueden requerir paquetes externos. Twisted [1] es una potente y bien organizada colección de módulos para añadir funcionalidades de red a los programas en Python.

Si estamos codificando un cliente de correo electrónico con soporte multi-protocolo (POP 3, SMTP, IMAP), el entorno de trabajo Twisted nos librará de tener que empezar desde cero. Twisted Mail incluye todos los protocolos importantes. Twisted tiene también módulos listos para usar de SSH, SFTP, HTTP (incluyendo HTTP/ 1.1), DNS, NNTP y Jabber. Si aún así queremos re-inventar la rueda implementando nuestros propios protocolos, los módulos *twisted.cred* y *twisted.spread* nos pueden ayudar a simplificar el trabajo.

Conectar Asíncronamente

Twisted es, básicamente, un marco de trabajo para comunicaciones asíncronas. Al contrario que otras librerías, las funciones de Twisted no provocan bloqueo

cuando se llaman. La aplicación continúa ejecutándose hasta que se le comunica que ya está disponible la información requerida. Aunque las librerías estándar de Python soportan esto (el módulo *asyncore* tiene funcionalidades básicas para conmutar entre múltiples canales I/O dentro de un hilo), Twisted implementa este diseño a un nivel mayor en sus protocolos, interfaces y componentes. Esto permite a los programadores escribir aplicaciones de red que funcionan sin necesidad de procesos e hilos adicionales, controlando al mismo tiempo múltiples canales I/O.

Altamente modular

Para crear con Twisted nuestro propio software orientado a red, tendremos que determinar en primer lugar qué parte del marco de trabajo necesitamos. Los desarrolladores dividieron Twisted en varios subproyectos cuando se pasó de la versión 1 a la 2, con el objetivo de añadir más claridad. La Tabla 1 tiene una lista de los elementos más importantes, y existe una lista completa en [2].

Tabla 1: Secciones Man

<code>twisted</code>	Marco de trabajo para aplicaciones asíncronas, la base de todos los subproyectos Twisted
<code>twisted.conch</code>	Implementación de los protocolos SFTP y SSH para clientes y servidores
<code>twisted.web</code>	Protocolo HTTP para clientes y servidores
<code>twisted.web2</code>	Soporte para el protocolo HTTP/ 1.1 como marco de trabajo del servidor. Este paquete está aún en desarrollo, y no debería usarse para aplicaciones críticas
<code>twisted.mail</code>	Implementación de los protocolos SMTP, IMAP y POP para clientes y servidores
<code>twisted.names</code>	Soporte para protocolo DNS para clientes y servidores
<code>twisted.news</code>	Protocolo NNTP para clientes y servidores
<code>twisted.words</code>	Módulo para chat o aplicaciones de mensajería instantánea

Twisted requiere el módulo Zope Interface, que implementa interfaces de los que carece el núcleo del lenguaje Python. Si sólo vamos a usar un único proyecto, como Twisted Web o Twisted Mail, podemos descargar el módulo requerido de la página del proyecto. Alternativamente, Twisted Sumo [3] tiene todos los módulos (estables), incluyendo una interfaz de Zope.

Tras descomprimir el paquete, tecleamos `python setup.py install`. Si la interfaz Zope no está instalada, el instalador mostrará un mensaje de error y terminará.

El módulo `twisted.cred` controla la autenticación en las comunicaciones cliente-servidor. Permite que múltiples protocolos se conecten al sistema, para autenticarse y para intercambiar información. Por ejemplo, el soporte para POP 3 en Twisted proporciona una combinación de nombre de usuario y contraseña para abrir el buzón de correo solicitado. El llamado Perspective Broker es de vital importancia aquí. El broker proporciona acceso a objetos remotos e implementa copia, referenciado y cacheo de objetos.

Conexión a Base de Datos

`twisted.enterprise` proporciona una interfaz de base de datos que es compatible con la DB-API 2.0 de Python. Esto hace del acceso a bases de datos MySQL, Oracle o PostgreSQL un juego de niños. El módulo usa una interfaz asíncrona, que puede ejecutarse en múltiples hilos sin perder la seguridad del hilo en un bucle principal basado en eventos de Twisted. El bucle principal tiene lugar dentro del módulo `twisted.internet`, conocido como el *reactor*. Implementa el bucle infinito del programa en el que Twisted controla varios eventos. El reactor proporciona la interfaz subyacente a las principales funcionalidades internas de Twisted, como las conexiones de red, manejo de hilos o control de eventos.

Otros módulos raramente son necesarios en aplicaciones prácticas, como `twisted.protocols` o `twisted.manhole`. Conch implementa la versión 2 del protocolo Secure Shell para Twisted. El documento "Cómo" existente en [5] explica la implementación de un cliente SSH con Conch en unos pocos y sencillos pasos.

Los campos de aplicación más interesantes para Twisted son probablemente

la Web y las aplicaciones para servidores, mediante el uso de `twisted.web` o `twisted.web2`. La potente plantilla para esta tarea se llama Nevow [6]. Este entorno de trabajo tiene también un conjunto completo de funciones necesarias para programar clientes HTTP. La API `twisted.web` soporta múltiples capas de abstracción: desde servidores web sencillos, pasando por soporte de sesiones, hasta servidores de aplicaciones interactivas y páginas web distribuidas: las posibilidades son numerosas.

Cuando una solicitud de un cliente llega a un servidor Web, el servidor crea un objeto de solicitud y lo controla hasta el sistema con los recursos, donde se crea la respuesta (véase Figura 1).

Además de `twisted.web`, tenemos también el módulo `twisted.web2`, que está aún en desarrollo. Las principales mejoras son las siguientes:

- Soporte para HTTP 1.1
- Filtros internos y de salida predefinidos, para servir páginas Web comprimidas con gzip
- Separación del manejo de peticiones de alto y bajo nivel.
- Análisis correcto de cabeceras HTTP.
- Reescritura enormemente mejorada de URL, si se usa en combinación con un proxy.

A pesar de que hay mejoras muy importantes en `twisted.web2` en comparación con la versión anterior, los desarrolladores no recomiendan usar el nuevo módulo de momento, entre otras cosas porque el módulo nuevo es más lento que su predecesor.

El último módulo de importancia es Twisted Mail, que implementa SMTP, POP3 e IMAP4. Además de estos protocolos, el módulo es capaz de leer y escribir en formato Maildir de buzones de correo. Existe también una combinación preconfigurada de SMTP y POP3 para servidores de correo y sistemas de hosting virtual. Y Twisted Mail entiende la mayoría de las opciones de Sendmail, que pueden ser útiles para aplicaciones compatibles de descarga.

Congelar el servidor

El marco de trabajo Twisted tiene una colección de programas en línea de comandos que preparan a las aplicaciones de Twisted para escenarios particulares [7]. Las herramientas `mktp` y `tapconvert` crean archivos en formato

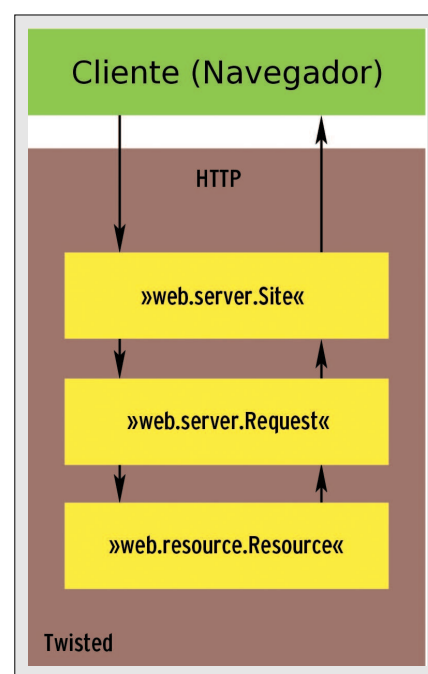


Figura 1: Proceso de petición Web dentro del marco de trabajo Twisted.

Tap, Tas o Tax desde el código fuente en Python. El código puede entonces usarse en los distintos servidores de Twisted, como Web, FTP o IRC.

La herramienta `twistd` devuelve la vida a los archivos Tap. Estrictamente hablando, `twistd` no es necesario para ejecutar las aplicaciones Twisted, pero hace las cosas más fáciles, ya que toma el control del reactor y controla el inicio y el apagado del programa. Además `twistd` admite la selección de un tipo de reactor diferente, permitiendo que una aplicación pueda ejecutarse en modo demonio o que escriba archivos de log. Los programas `tap2deb` y `tap2rpm` comprimen las aplicaciones de servidor terminadas para su distribución en paquetes con formato Debian o RPM. Las herramientas generan automáticamente scripts para la instalación y la eliminación del servidor.

El Listado 1 muestra un servidor Web sencillo y demuestra la potencia de Twisted. Podríamos usar este código para habilitar la configuración basada en Web de nuestra aplicación. Las dos primeras líneas cargan los módulos requeridos. La línea 6 crea un servidor nuevo con un directorio raíz para los documentos HTML `/var/www/htdocs` en nuestro ejemplo. La siguiente instrucción le indica al reactor que se quede a la escucha de peticiones en el puerto 7777. La llamada `reactor.run()` inicia el servidor Web.

El Listado 2 muestra un servidor Web que añade un buen número de opciones al marco de trabajo básico del Listado 1.

Listado 1: Servidor Web Sencillo

```
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()
```

En primer lugar, las líneas 9 a 11 habilitan el soporte para scripts Perl: queremos que el servidor entregue los archivos con extensión `.pl` al intérprete Perl, `/usr/bin/perl` y que devuelva el resultado. El soporte para PHP y otros lenguajes puede habilitarse fácilmente en este punto.

El método `putChild()` de la línea 14 fija el directorio CGI a `/var/www/cgi-bin`, permitiendo de esta manera el acceso a los scripts en `http://servername:7777/cgis`. El método especifica también la ruta del directorio `doc`, `/var/www/doc`. La variable `indexNames` especifica qué archivos busca el servidor en una petición al directorio. La orden se define por los nombres de archivo especificados.

Las siguientes herramientas de Twisted podrían usarse como alternativa al código del servidor Web:

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

`mktap` crea un archivo Tap preconfigurado. En este caso, `mktap` fija el directorio raíz a `/var/www/htdocs` y el puerto a 7777. El resultado acaba en el archivo `web.tap`, que es usado como parámetro por el servidor `twisted`. Tras el arranque, el servidor, el archivo de logs, `twistd.log` y el archivo PID (que puede usarse para matar al servidor con `kill 'cat twistd.pid'`) se ubican en el directorio en uso. `mktap web -help` nos ofrece más información de las opciones disponibles.

Sencillo Pero Potente

Este artículo nos da sólo una fugaz visión general de Twisted. Este marco de trabajo, junto con Twisted Mail, Conch, Twisted Web(2) y los demás subproyectos, forman la base de muchas aplicaciones de red comerciales. Hasta la NASA usa Twisted Matrix [8].

Ayudar a las aplicaciones en Python a que alcancen las estrellas hace la vida mucho más fácil a los saturados programadores aquí en la Tierra. Twisted reduce claramente el esfuerzo que supone desarrollar un cliente o servidor eliminando la necesidad de implementar protocolos conocidos. Esto acerca a los programadores aplicaciones en red seguras, rápidas, estables y flexibles, a un golpe de ratón. ■

Listado 2: Servidor Web Ampliado

```
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()
```

RECURSOS

- [1] Página del proyecto Twisted Matrix: <http://www.twistedmatrix.com>
- [2] Vista general de todos los proyectos Twisted: <http://www.twistedmatrix.com/projects>
- [3] Twisted Sumo: <http://twistedmatrix.com/projects/core>
- [4] Interfaz Zope: <http://www.zope.org/Wikis/Interfaces>
- [5] Cliente SSH con Conch: http://twistedmatrix.com/projects/conch/documentation/howto/conch_client.html
- [6] Nevow (sistema de plantillas): <http://divmod.org/projects/nevow>
- [7] Herramientas: <http://twistedmatrix.com/projects/core/documentation/howto/basics.html>
- [8] Usuarios de Twisted: <http://twistedmatrix.com/services/success>