thor
thunder
tronar
tronada
tornar
tornada
tornada
tronada
+

tornada
what is it?

Non-blocking web server
Based on epoll / kqueue
Handles 1000s of connections
Powers FriendFeed
why do you even care?

Know what you use

I can't trust what I don't know

Learning and sharing: fun!
what do you use it for?

We built a

Scalable
Distributed
Fault tolerant
High load
[CITATION NEEDED]
oscar.vilaplanan@paylogic.eu
TORNADO
what is it made of?

IOLoop

Callbacks, Tasks, Timeouts

TCPServer

Application

RequestHandler

Generators
can I extend it?

Sure!

It’s easy

How would we make a simple TCPServer?
show me the source!

0b_tcpsserver.py
GETTING STARTED
how do I make the simplest app?

Define a RequestHandler

Implement get

Define an Application

Tell the application to listen

Start the IOLoop
show me the source!

01_getting_started.py
what is an Application?

Collection of RequestHandlers

Implements listen:

Starts an HTTPS server

Sets the Application as request callback

Implements __call__:

Handles the user requests
show me the source!

t1_application_listen.py
what does Application.__call__ do?

Parses the URL
Decides which handler to use
and creates an instance of it
(each connection gets one)

_executes it
(passing any defined transforms to it - e.g. Gzip compression)
show me the source!

t2_application_call_call.py
what does RequestHandler._execute do?

Calls the handler method

Checks XSRF cookie

Maybe closes the connection
show me the source!

t3_request_handler___execute.py
what is the IOLoop?

Core of Tornado
Used for server and client
Usable standalone or with WSGI
Single instance per process
Small!
**what does the IOLoop do?**

Loops forever

Executes:

- Callbacks (asap)
- Timeouts (when due)
- Events (when occurred)
**what is an Event?**

Something that **happened** on a socket (fd)

(e.g. a user opened a connection)

**Applications** define **handlers** for Events
how do I wait for an Event?

add_handler(fd, handler, events)
update_handler(fd, handler, events)
remove_handler(fd, handler)

“Notify me when I can READ or WRITE, or when there is an ERROR”
IOLOOP

THE LOOP
IN FOUR STEPS
Process the Callbacks

They were scheduled by previous:

Callbacks

Event handlers
For each Timeout:

Is it due now?

Run its callback

Calculate the \textit{time} till next Timeout
Poll timeout:

Callbacks? Then 0

Timeouts? Then time until the next Timeout

Neither? 1 hour

Here the IOLoop blocks
and fourth process the Events

For each (file descriptor, Event):

Call its handler

...and repeat the loop
THE EXAMPLE

WHAT IT REALLY DOES
show me the source!

01_getting_started.py
what does the example do?

Application.listen starts HTTPServer

calls IOLoop.add_accept_handler

(Application.__call__ will handle the ACCEPT event)

and when a client connects...
what does the example do?

- connection polls
- IOLoop calls
- TCPServer which is HTTPServer

- headers read and calls
  - body using callbacks

- request_callback
  - Application.__call__
    - (after parsing URL)
      - RequestHandler._execute

- HTTPConnection writes to (in chunks)

- get or post or ...
  - RequestHandler

- _auto_finish? finishes
  - yes
show me the source!

t4_simple_ioloop.py
SCHEDULED TASKS
how do we schedule a task?

Ioloop.add_callback

Ioloop.add_timeout

PeriodicCallback

Call me back asap

Call me back later

Call me often
show me the source!

02_scheduled_tasks.py
how does `add_callback` work?

Adds the callback to the list of callbacks.

(and wraps it in a threadlocal-like stack context)
what do you mean, threadlocal-like?

StackContext

Keeps track of the socket connection

Handles association between socket and Application classes
show me the source!

t6_add_callback.py
how does add_timeout work?

Pushes the timeout to the heap of timeouts.

(and wraps it in a threadlocal-like stack context too)
show me the source!

t7_add_timeout_timeout.py
how do PeriodicCallbacks work?

Schedules the next `timeout` to call `_run`
Marks the PeriodicCallback as `running`

Removes the next `timeout`
Marks the PeriodicCallback as `stopped`

Calls the `callback`
(unless `stop` was called)
show me the source!

t8_periodic_callback.py
what about Callback?

Indeed.
ASYNC HANDLERS

@ASYNCHRONOUS & _AUTO_FINISH
how does @asynchronous work?

Sets _auto_finish to False
(and does some Exception wrapping)

The connection remains open after get, post...

Close it yourself (whenever you want)
show me the source!

03_fetch_async.py
I put a callback on your callback

Nested callbacks make ugly code.
what about Callback?

Indeed.
GENERATORS
how do I avoid callbacks?

Use Callback (finally!)

and yield

Or Task
show me the source!

04_fetch_gen.py
YIELD POINTS
what is a YieldPoint?

Something you yield

Then stuff happens
what is a YieldPoint?

Callback
Sends a result for a key

Wait
Waits till a result for a key arrives

WaitMany
Same as Wait, for many keys
what is a YieldPoint?

Wait + Callback
(with an auto-generated key)

List of YieldPoints

Task

Multi
how do we do async processing?

callback=(yield Callback("key"))

When a result arrives, send it for the key “key”
how do we do async processing?

response=yield Wait(“key”)

When the result is sent, read it into response.
show me the source!

04_fetch_gen.py
show me the source!

05_task.py
show me the source!

t9_yield_points.py
how do we use websockets?

Extend WebSocketHandler (instead of RequestHandler)

Implement on_message
show me the source!

06_websocket.py
how do websockets work?

Similar to @asynchronous

(the connection is kept open)

After writing, read for more

(asynchronously—see IOStream)

To Application, it looks like a RequestHandler
how does WebSocket work?

Accepts the connection
Decides the version of the protocol.
Instantiates a WebSocketProtocol class
how does WebSocketProtocol work?

- **accept_connection**
  - Sends the required initial message
  - Reads the next message (asynchronously)

- **_write_response**
  - Writes a response on the socket
  - Reads the next message (asynchronously)
show me the source!

ta_websocket.py
iostream
What does IOStream do?

Communicates with the socket

Asynchronous

Uses IOLoop Callbacks

Let's fetch and return a webpage.
how does IOStream work?

_add_io_state

"Notify me when I can READ or WRITE, or when ERROR" schedules Callback for an event (READ, WRITE, ...)

_handle_events

Can I read? Call _handle_read
Can I write? Call _handle_write
Handles errors
how does IOStream work?

_store data in read buffer_
_Call the read callback (_read_from_buffer)_

_store data in read buffer_
_Call the read callback (_read_from_buffer)_

_write data from the buffer into the socket (handling funny circumstances)_
_Call the write callback (if it exists)
**how does IOStream work?**

socket.connect

_add_io_state(WRITE)

Set callback

Data in read buffer? Return it

Read buffer empty? _add_io_state(READ)

Add data to write buffer

_add_io_state(WRITE)
how do I use IOStream directly?

read_until_regex
read_until
read_bytes
read_until_close

All take a callback
how do I use streaming callbacks?

read_bytes
read_until_close

Param: streaming_callback

Data is sent to callback as it arrives
show me the source!

0a_async_callback.py
DATABASE
**how do I talk to a database?**

```python
import database

connection = database.connection

query("SQL")
```

Returns an iterable

Very simple
show me the source!

07_db.py
ASYNCMONGO
what is asyncmongo?

Asynchronous MongoDB client

Uses Tornado’s IOLoop
how do I use asyncmongo?

asyncmongo.Client

db.find

takes a callback parameter
show me the source!

08_mongo.py
**how does asyncmongo work?**

- Implements a `Connection`
- Many: `ConnectionPool`
- Sends data via Tornado’s `IOStream`
- Commands send via `Cursor`
- `Cursor.send_message`
- Uses `Connection` to communicate asynchronously

Let’s fetch and return a webpage.
can we write our own asyncmysql?

Difficult

C driver has blocking calls

mysql_query
mysql_store_result
mysql_use_result

Alternative

Use Twisted’s txMySQL with tornado.platform.twisted

Slower
Q & A
GRAZIE!

dev@oscarvilaplanacat