

ZeroMQ + Tornado

Aquí tenéis una demo que acabo de montar de un PUB\SUB con ZeroMQ + Tornado + WebSockets.

Tornado publica por cada websocket abierto lo que recibe por el socket ZMQ haciendo así de Streamer donde los receptores son los clientes JS activos.

Ésta es la clase WebSocket (fichero: TornadoWSock.py) la cual, al abrirse una conexión, se subscribe al puerto ZPORT. Se crea un Streamer y se le indica que su función de callback a la hora de recibir será send_to_clients, cuando se reciba un mensaje será a esa a la que se llamará la cual repitará dicho mensaje por su correspondiente conexión abierta.

```
class WebSocket(tornado.websocket.WebSocketHandler):
    clients = set()
    def open(self):
        self.clients.add(self)
        subscriber = context.socket(zmq.SUB)
        subscriber.connect("tcp://127.0.0.1:%s" % str(ZPORT))
        subscriber.setsockopt(zmq.SUBSCRIBE, '')
        self.subscribe_stream = ZMQStream(subscriber)
        self.subscribe_stream.on_recv(self.send_to_clients)

    def on_message(self, message):
        pass

    def send_to_clients(self, message):
        self.write_message(unicode(message))

    def on_close(self):
        self.clients.remove(self)
        self.subscribe_stream.close()
```

En el fichero Sender.py tenemos un socket publicando el texto introducido por consola al mismo ZPORT.

Para instalar las dependencias:

```
$ pip install -r requirements.txt
```

Lanzar el servidor:

```
$ python TornadoWSock.py
```

Abrir navegador a <http://127.0.0.1:8888/index.html>.

Lanzar el sender:

```
$ python Sender.py
```

Código: [zmq_tornado_websockets.zip](#)

Nota:

Para realizar un chat basado en websockets utilizando este código sería muy sencillo. Únicamente deberíamos colocar un socket PUB, como el de Sender.py, de forma global en TornadoWSock.py, y en el on_message de la clase WebSocket llamaríamos al send de ese socket con el mensaje recibido.