Tornado Cheat Sheet

Async · WebSocket · IOLoop · Non-Blocking I/O

SETUP & APPLICATION

```
pip install tornado
import tornado.web, tornado.ioloop
app = tornado.web.Application([
(r'/hello', HelloHandler),
(r'/ws', WebSocketHandler),
])
app.listen(8888)
tornado.ioloop.IOLoop.current().start()
```

REQUEST HANDLERS

```
class
MainHandler(tornado.web.RequestHandler):
async def get(self):
name = self.get_argument('name',
'World')
self.write(f'Hello {name}')
async def post(self):
data =
tornado.escape.json_decode(self.request.body)
self.set_status(201)
self.write(data)
```

ROUTING

```
app = tornado.web.Application([
(r'/', MainHandler),
(r'/users/(?P<user_id>[0-9]+)',
UserHandler),
(r'/static/(.*)',
tornado.web.StaticFileHandler,
{'path': 'static'}),
])
```

URL patterns are regexes; groups become positional args

ASYNC HANDLERS

```
class
AsyncHandler(tornado.web.RequestHandler):
async def get(self):
result = await some_async_function()
self.write({'result': result})
# HTTP fetch:
http =
tornado.httpclient.AsyncHTTPClient()
response = await
http.fetch('http://example.com')
```

WEBSOCKETS

```
class
ChatHandler(tornado.websocket.WebSocketHandler):
def open(self):
clients.add(self)
def on_message(self, msg):
for c in clients: c.write_message(msg)
def on_close(self):
clients.discard(self)
# JS: ws = new WebSocket('ws://host/ws')
```

TEMPLATES

```
class
PageHandler(tornado.web.RequestHandler):
def get(self):
self.render('index.html', title='Hi',
items=[])
# index.html:
{% for item in items %}{{ item }}{% end
%}
{{ escape(user_input) }}
```

Auto-escaping enabled by default

REQUEST & RESPONSE

```
self.request.method / .headers / .body
self.get_argument('key', default=None)
self.get_body_argument('key')
self.request.files['file'][0]['body']
self.set_header('Content-Type',
'application/json')
self.set_status(200)
self.write(json.dumps(data))
```

COOKIES & SESSIONS

```
self.set_cookie('user_id', str(user.id),
httponly=True)
self.get_cookie('user_id')
# Secure signed cookie:
self.set_secure_cookie('user_id',
str(user.id))
self.get_secure_cookie('user_id')
cookie_secret='random-secret-key' in
Application settings
```

IOLOOP BASICS

```
loop = tornado.ioloop.IOLoop.current()
loop.run_sync(main_coroutine)
# Periodic callback:
cb =
tornado.ioloop.PeriodicCallback(check,
5000)
cb.start()
# Schedule once:
loop.call_later(30, one_time_task)
```

Single-threaded event loop; never block the loop

ERROR HANDLING

```
def write_error(self, status_code,
**kwargs):
self.write({'error': self._reason})
raise tornado.web.HTTPError(404, 'Not
found')
raise tornado.web.Finish()  # stop
handler, no error
# Unhandled exceptions return 500
```

TESTING

```
from tornado.testing import
AsyncHTTPTestCase
class Test(AsyncHTTPTestCase):
def get_app(self): return app
def test_get(self):
resp = self.fetch('/hello')
self.assertEqual(resp.code, 200)
self.assertIn(b'Hello', resp.body)
```

COMMON GOTCHAS

Blocking I/O in handlers stalls all requests
Use run_on_executor for CPU-bound code
WebSocket check_origin() returns False by default for XSS
Exceptions in callbacks must be handled or they're swallowed
Don't use time.sleep() use asyncio.sleep()