Связываем синхронный фреймворк с асинхронным (на примере django)

Михаил Коробов, 4 июня 2011

http://www.devconf.ru

Хотим, чтобы некоторые части сайта работали в режиме реального времени. Что делать?

- Переписать все на node.js, конечно;
- переписать все на twisted;
- переписать на tornado;
- erlang?
- ...
- ?

А почему по-старинке-то нельзя?

А почему по-старинке-то нельзя?

Можно.

Но не всегда.

Transparent Adaptation: не обязательно все переписывать

Transparent Adaptation: не обязательно все переписывать

- Однопользовательский/«синхронный» код писать проще;
- к тому же он часто уже написан;
- можно использовать технологии, больше подходящие к задаче;
- клиентские технологии все еще ненадежны.

Transparent Adaptation: не обязательно все переписывать

- Однопользовательский/не-реалтайм код писать проще;
- к тому же он часто уже написан;
- можно использовать технологии, больше подходящие к задаче;
- клиентские технологии все еще ненадежны.

Progressive Enhancement

Технологии, с помощью которых можно прикрутить realtime-фичи к django-сайту

- node.js
- twisted
- opencoweb
- gevent
- eventlet
- hookbox
- APE
- tornado
- pusher.com
- StreamHub
- meteorserver.org
- orbited
- xmpp + strophe.js
- и т.д.

Технологии, с помощью которых можно прикрутить realtime-фичи к django-сайту

- **node.js**
- twisted
- opencoweb
- gevent
- eventlet
- hookbox
- APE

- **tornado**
- pusher.com
- StreamHub
- meteorserver
- orbited
- xmpp + strophe.js
- и т.д.

Пример: django + node.js

Преимущества node.js

- Там ничего сложного;
- куча готовых асинхронных библиотек;

Преимущества node.js

- Там ничего сложного;
- куча готовых асинхронных библиотек;
- любопытно попробовать.

django + node.js: как?

Django слушает 80 порт, node.js – какой-то другой.

Django знать про node.js почти ничего и не нужно.

Редактор текста? Жми “сохранить”.

Чат? Джанга совсем не при чем.

django + node.js: устанавливаем node.js в virtualenv

```
workon myenv
cd node-v0.4.8
./configure –prefix=“/path/to/myenv/”
make
make install
```

django + node.js: устанавливаем npm в virtualenv

```
workon myenv
git clone git://github.com/isaacs/npm.git
cd npm
make
make install
```

Пишем чат на socket.io: установка

```
npm install socket.io
```

Пишем чат на socket.io: сервер

```
var http = require('http'), io = require('socket.io');

server = http.createServer();
server.listen(8080);
var socket = io.listen(server);

socket.on('connection', function(client){
    client.on('message', client.broadcast);
});
```

Пишем чат на socket.io: клиент (html)

```
<html>
<head><script src="http://my-server.com:8080/socket.io/
   socket.io.js"></script></head>
<body>
<div id='messages'></div>
<input type="text" id='msg'>
<input type="button" onclick="send();” value="Send">
</body>
</html>
```

Пишем чат на socket.io: клиент (js)

```
var socket = new io.Socket(null, {port: 8080});
var $ = document.getElementById;
socket.on('message', function(msg){
    $('messages').innerHTML += msg + '<br>';
});
socket.connect();
function send(){
    socket.send($('msg').value);
}
```

Пишем чат на socket.io: запуск

```
$ node app.js

; или через supervisord, например
[program:node]
command = {{ ENV_DIR }}/bin/node {{ PROJECT_DIR }}/app.js
stderr_logfile = {{ ENV_DIR }}/var/log/node.error.log
stdout_logfile = {{ ENV_DIR }}/var/log/node.log
```

django + node.js: проблемы

- Авторизация;
- потенциальное дублирование логики.

django + node.js:
пути решения проблем

- Общение через общее хранилище данных (не всегда удобно – например, как распиклить данные из джанговских сессий?);

- общение через API.

django + node.js: уроки

- Серверная часть для realtime-штук может быть совсем примитивной – держать соединения и раздавать сообщения;
- большая часть логики может быть на клиенте;
- можно повторно использовать логику из синхронной части сайта, сделав небольшой API, и обращаясь к нему с асинхронного сервера.

Вся логика на клиенте, примитивный сервер + API-хуки – см. hookbox.org, pusher.com, stream-hub.com

django + tornado

django + tornado: преимущества

- Один язык для серверной части;
- возможность повторного использования кода без создания API;
- больше библиотек общего назначения.

django + tornado: недостатки

- меньше асинхронных библиотек (в node.js они все асинхронные, а тут нужно быть внимательным);
- применительно к socket.io: связка поддерживается не автором socket.io (что, впрочем, и к лучшему) и может поломаться при смене протокола;
- ~~скорость~~.

django + tornado:
почему не twisted, gevent, ...?

Просто так.

django + tornado: как?

- tornado запускаем как management-команду django;
- socket.io + tornadio;
- используем джанговские средства для авторизации;
- опционально (но удобно) - redis (через brukva);

django + tornadio: запускаем как tornado как management-команду django

```
from my_realtime_app import MyConnection

class Command(NoArgsCommand):
    def handle_noargs(self, **options):
        application = tornado.web.Application(
            [tornadio.get_router(MyConnection)],
            socket_io_port = 8080)
        tornadio.server.SocketServer(application)
```

django + tornadio: авторизация (клиент)

```
socket.on('connect', function(){
    socket.send({type: ‘auth request’,
        sessionid: Cookie.read('sessionid')
   });
});
socket.on('message', function(msg){
    if (msg.type != 'auth result’) return;
    connection.isAuthorized = msg.authorized;
});
```

django + tornadio: авторизация (сервер)

```
class MyConnection(tornadio.SocketConnection):
    def on_message(self, message):
        if message.type != ‘auth request’:
            return
        self.session = get_session(message.sessionid)
        self.user = get_user(self.session)
        self.send({'type': 'auth result',
            ’authorized': self.user.is_authenticated()
        })
```

django + tornadio: авторизация (сервер)

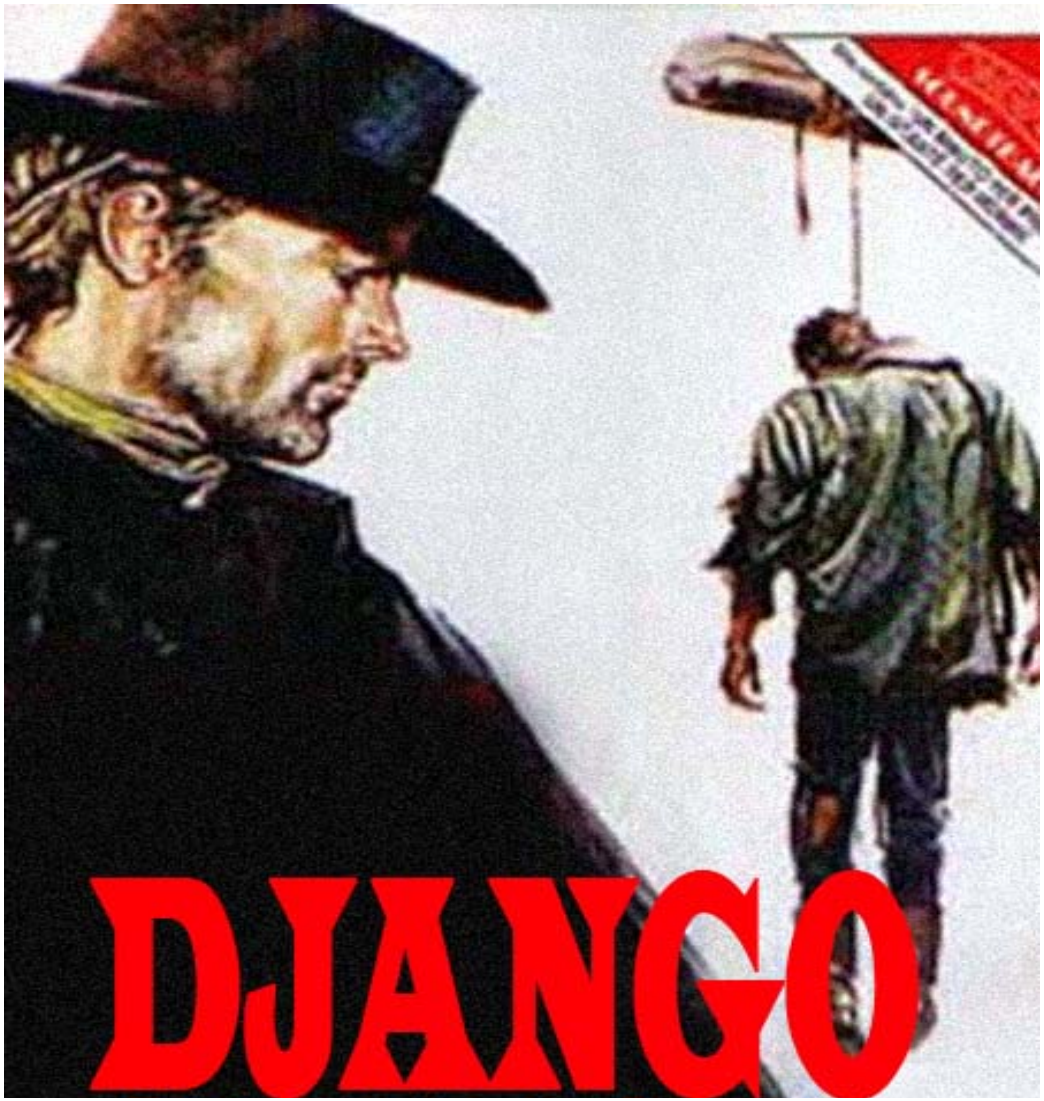
```
from django.utils.importlib import import_module
_engine = import_module(settings.SESSION_ENGINE)

def get_session(session_key):
    return _engine.SessionStore(session_key)
```

django + tornadio: авторизация (сервер)

```
def get_user(session):
    class Dummy(object):
        pass
    django_request = Dummy()
    django_request.session = session
    return django.contrib.auth.get_user(django_request)
```

Эй, стоп! А как же блокировки?
django ORM синхронный ведь?


DJANGO

Использование синхронного кода django в tornado-части: варианты

- «Психологическое решение»;
- работать с базой через асинхронные библиотеки, минуя ORM;
- сделать API;
- выносить работу в треды;
- для ковбоев – все запатчить через gevent/eventlet.

Использование синхронного кода django в tornado-части: “психологическое решение”

- Питонья логика бывает и без i/o;
- блокировки и так неизбежны (как минимум на время вычислений);
- i/o бывает достаточно быстрым (та же аутентификация);
- та же авторизация требуется 1 раз при установке соединения, а не на каждое сообщение;
- Ben Darnell (автор tornado): “если обработка запросов задерживается из-за работы с базой, то сервер в любом случае не справляется с нагрузкой”

Использование синхронного кода django в tornado-части: “психологическое решение”

```
ioloop.set_blocking_log_threshold(seconds)
```

```
yappi.start()
```

Использование синхронного кода django в tornado-части: API

- Самое простое - http API;
- ~~дублирование логики~~;
- синтаксический сахар (пример: tornado-slacker).

Использование синхронного кода django в tornado-части: API

```
def on_message(self, message):
    # …
    data = MyModel.objects.filter(foo__gt=F('bar'))
    # …
```

Использование синхронного кода django в tornado-части: API

```
@adisp.process
def on_message(self, message):
    # …
    data = yield MyModel.objects.filter(foo__gt=F('bar'))
    # …
```

Использование синхронного кода django в tornado-части: API

```
from slacker import Slacker
from slacker.workers import DjangoWorker
from my_app.models import MyModel as _MyModel

MyModel = Slacker(_MyModel, DjangoWorker())
```

Использование синхронного кода django в tornado-части: API

```
from slacker import Slacker
from slacker.workers import ThreadWorker
from my_app.models import MyModel as _MyModel

MyModel = Slacker(_MyModel, ThreadWorker())
```

Спасибо.