HTTP API & Py

Milan Čermák
Pyvo Brno,
31. 1. 2013

Why API?

APIs lead to success

What makes a good API?

Consistency

Consistency

Predictability

Consistency

Predictability

Adherence to standards

Consistency

Predictability

Adherence to standards

Great documentation

Why HTTP?

"While HTTP isn't always the best answer, it's a damn fine first guess."

Coda Hale

Benefits of HTTP

Benefits of HTTP

The most widespread application protocol

Benefits of HTTP

The most widespread application protocol

Statelessness

Benefits of HTTP

The most widespread application protocol

Statelessness

Optional caching

Benefits of HTTP

The most widespread application protocol

Statelessness

Optional caching

Promotes layered infrastructure

Benefits of HTTP

The most widespread application protocol

Statelessness

Optional caching

Promotes layered infrastructure

etc.

Limitations of HTTP

Limitations of HTTP

Authorization

Limitations of HTTP

Authorization

Statelessness

Limitations of HTTP

Authorization

Statelessness

Verbosity

Limitations of HTTP

Authorization

Statelessness

Verbosity

Crippled parallelism

```
"REST" if protocol.startswith("http") else "you're doing it wrong"
```

Set of architectural constraints

Set of architectural constraints

Client-server

Set of architectural constraints

Client-server

Stateless

Set of architectural constraints

Client-server

Stateless

Cacheable

Set of architectural constraints

Client-server

Stateless

Cacheable

Layered

Set of architectural constraints

Client-server

Stateless

Cacheable

Layered

Uniform interface *

How can Python help?

OOP

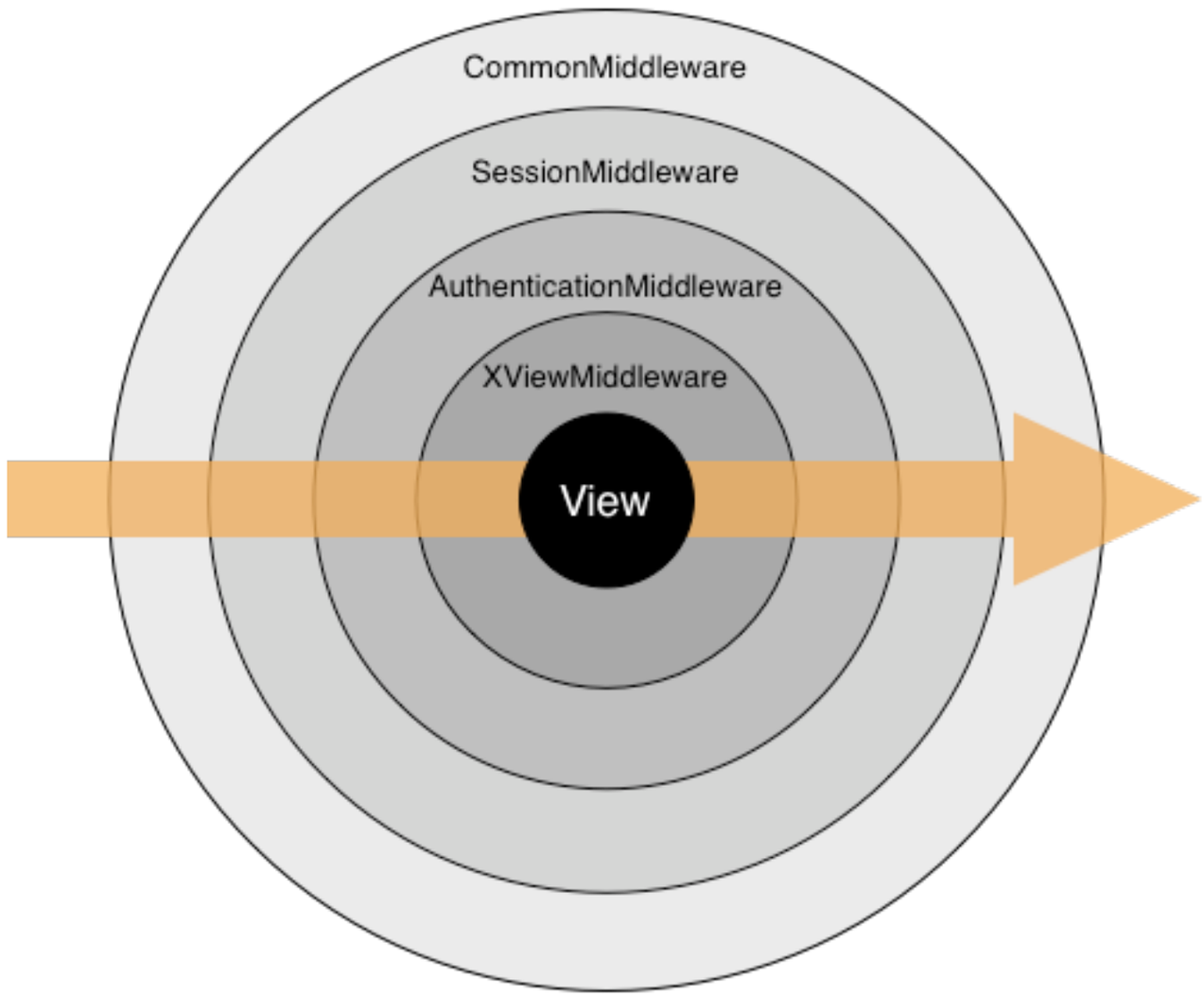
OOP

URI ~ Class handler mapping

```
import handlers

urls = [(r"/user", handlers.users.NewUser),
        (r"/user/(\d+)", handlers.users.User)]

class User(handler.base.BaseHandler):
    def delete(self, user_id):
        pass
    def get(self, user_id):
        pass
    def post(self, user_id):
        pass
```

```
class UserValidatorMixin(object):
    def check_user_data(self, user_dict):
        pass

class User(handler.base.BaseHandler,
           UserValidatorMixin):

def post(self, user_id):
    new_user = self.get_argument(“user”)
    if not self.check_user_data(new_user):
        return self.http_error(400, “Invalid data”)
```

Middleware


CommonMiddleware
SessionMiddleware
AuthenticationMiddleware
XViewMiddleware
View

Extending JSONEncoder

```
import json

class AppJSONEncoder(json.JSONEncoder):

    def default(self, obj):
        if isinstance(obj, User):
            return {"name": obj.name,
                    "height": obj.height,
                    "cash": obj.get_bank_account_balance()}
        return json.JSONEncoder.default(self, obj)

user = User("1337")
json.dumps(user, cls=AppJSONEncoder)
```

What about mobile?

Compression of HTTP bodies

```
GET /user/1337 HTTP/1.1

Host: api.napyvo.io

Accept-Encoding: gzip, identity
```

Compression of HTTP bodies

```
HTTP/1.1 200 OK

Content-Encoding: gzip

Content-Type: application/json; charset=utf-8
```

Compression of HTTP bodies

```
POST /user HTTP/1.1

Host: api.napyvo.io

Content-Encoding: gzip

Content-Type: application/json


[gzipped representation of a user]
```

Caching

```
Cache-Control: max-age=3600

Expires: Thu, 31 Jan 2013 22:00:00 GMT
```

```
<- Last-Modified: Thu, 31 Jan 2013 18:30:00 GMT

-> If-Modified-Since: Wed, 30 Jan 2013 13:37:00 GMT
```

```
ETag: foo

If-None-Match: foo
```

Partial resources

```
GET /car/9


{"car": {
    "color": "red",
    "passengers": [
        {"href": "/user/1337",
         "rel": "self"}]
    }
}
```

```
GET /car/9?zoom=passengers

{"car": {

    "color": "red",

    "passengers": [

        {"href": "/user/1337",

         "rel": "self",

         "name": "Milan",

         "drink": "beer",

         "skills": ["python", "http"],

         }]

    }

}
```

The promise of Hypermedia

Hypermedia as the engine of application state

Q & A