WSGI, Werkzeug
And the challenges of building a web framework

@igorgue

The Problems

- A paradigm shift in web applications
- More JavaScript! (about 40% of our code base)
- Applications are getting small
- Backend is no longer so tied to the front end (no more crazy render of templates in the backend)

I want to write a Framework

- Make it just talk REST
- No ORM
- No templates
- Just a routing mechanism
- http://pappoproject.org/ (started as a Haskell project)

My Goal

http://webmachine.basho.com/images/http-headers-status-v3.png

Hello, World! Time

WSGI

```
def hello_world(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain')])
    return ["Hello, World!"]
```

Run it… but don’t

```
if __name__ == '__main__':
    from wsgiref.simple_server import make_server
    host = 'localhost'
    port = 8000
    server = make_server(host, port, hello_world)

    print "Server running on http://{host}:{port}".format(host=host,
port=port)
    server.serve_forever()
```

Use a Webserver!

mod_wsgi

Gunicorn

uWSGI

Gunicorn*

```
$ gunicorn module:main_function_or_class
```

Or in our example:

```
$ gunicorn hello_world:hello_world
# if we have a hello_world.py module
```

* my personal favorite

Parsing Query String

```
def hello_world(environ, start_response):
    parameters = parse_qs(environ.get('QUERY_STRING', ''))

    if 'subject' in parameters:
        subject = escape(parameters['subject'][0])
    else:
        subject = "World"

    start_response('200 OK', [('Content-Type', 'text/plain')])

    return ["Hello, {subject}!".format(subject=subject)]
```

Dealing with the Path

```
def hello_world(environ, start_response):
    requested_path = environ['PATH_INFO']
    parameters = parse_qs(environ.get('QUERY_STRING', ''))

    if requested_path == '/':
        if 'subject' in parameters:
            subject = escape(parameters['subject'][0])
        else:
            subject = "World"

        return ["Hello, {subject}!".format(subject=subject)]
    elif requested_path == '/lol' or requested_path == '/lol/':
        start_response('200 OK', [('Content-Type', 'text/plain')])

        return ["LOL!!!"]
    else:
        start_response('404 NOT FOUND', [('Content-Type', 'text/plain')])

        return ["Upps, your requested path '{path}' was not
found.".format(path=requested_path)]
```

Better Url Handling

```
def hello(environ, start_response):
    """Says hello to the user"""
    parameters = parse_qs(environ.get('QUERY_STRING', ''))

    if 'subject' in parameters:
        subject = escape(parameters['subject'][0])
    else:
        subject = 'World'

    start_response('200 OK', [('Content-Type', 'text/plain')])

    return ["Hello, {subject}!".format(subject=subject)]

def lol(environ, start_response):
    """Just prints 'LOL!!!'"""
    start_response('200 OK', [('Content-Type', 'text/plain')])

    return ["LOL!!!"]

def not_found(environ, start_response):
    """Shows a 404"""
    requested_path = environ['PATH_INFO']
    start_response('404 NOT FOUND', [('Content-Type', 'text/plain')])

    return ["Resource {path} couldn't be found".format(path=requested_path)]
```

Better Url Handling Matching

```
urls = [
    (r'^$', hello),
    (r'^lol/?$', lol)
]

def application(environ, start_response):
    """
WSGI application that will get served.
"""
requested_path = environ['PATH_INFO'].lstrip('/')

for regex, callback in urls:
    match = re.search(regex, requested_path)

    if match:
        return callback(environ, start_response)

return not_found(environ, start_response)
```

Werkzeug

```
from werkzeug.wrappers import Request, Response

def hello_world(environ, start_response):
    request = Request(environ)
    response = Response("Hello {0}".format(request.args.get('name',
"World")))

    return response(environ, start_response)
```

Routes

```
self.url_map = Map([
    Rule('/', endpoint='index'),
    Rule('/<name>', endpoint='dashboard'),
    Rule('/<name>/info', endpoint='contact_information')
])
```

User Agent Queries

```
from werkzeug.wrappers import Response
from werkzeug.useragents import UserAgent

def application(environ, start_response):
    browser = UserAgent(environ).browser # chrome
    response = Response("Hello {browser}".format(browser=browser))

    return response(environ, start_response)
```

Debugger Support

NameError

NameError: global name 'Response' is not defined

Traceback (most recent call last)

File "/Users/igor/code/wsgi_werkzeug/werkzeug/browser.py", line *6*, in application

```
response = Response("Hello {browser}".format(browser=browser))
```

NameError: global name 'Response' is not defined

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Brought to you by **DON'T PANIC**, your friendly Werkzeug powered traceback interpreter.

Debugger Shell

Traceback (most recent call last)

File "/Users/igor/code/wsgi_werkzeug/werkzeug/browser.py", line *6*, in `application`

```
  response = Response("Hello {browser}".format(browser=browser))
```

```
[console ready]
>>> response
```

Traceback (most recent call last):

File "<debugger>", line *1*, in `<module>`

```
  response
```

NameError: name 'response' is not defined

```
>>> print "Testing"
Testing
>>>
```

NameError: global name 'Response' is not defined

Deployment

```
<VirtualHost *>
    ServerName browserstuff.dev

    WSGIDaemonProcess browserstuff user=user1 group=group1
processes=2 threads=5
    WSGIScriptAlias / /Users/igor/code/wsgi_werkzeug/
werkzeug/app.wsgi

    <Directory /Users/igor/code/wsgi_werkzeug/werkzeug/>
        WSGIProcessGroup browserstuff
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

Deploy Gunicorn with Supervisord

```
[program:gunicorn]
command=/path/to/gunicorn main:application -c /path/to/
    gunicorn.conf.py
    directory=/path/to/project
    user=nobody
    autostart=true
    autorestart=true
    redirect_stderr=True
```

https://github.com/benoitc/gunicorn/blob/master/examples/supervisor.conf

Recap

- WSGI isn't hard
- Werkzeug gives you a lot of the base
- Stuff that Django doesn't even have
- Make your dreams come true, and you might be the next DHH

Thanks!

SENZARI

http://senzari.com we're hiring!

H<CKD>Y FOUNDATION
Presented by Florida International University

http://hackdayfoundation.org/ 2K prize!

Steal this:
http://igorgue.com/presentations/wsgi-werkzeug.pdf