WSGI

Основы Веб-программирования

Кафедра Интеллектуальных Информационных Технологий, ИнФО, УрФУ

Ресурсы

https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface

http://lectureswww.readthedocs.org/5.web.server/wsgi.html

WSGI - это…?

Python
pep-333
pep-3333

Спецификация простого и универсального интерфейса между Веб-сервером, веб-приложением или Веб-фреймворком.

Кто использует?

Все кто на Питоне

BlueBream, bobo, Bottle, CherryPy, Django, Eventlet, Flask, Gevent-FastCGI, Google App Engine’s webapp2, Gunicorn, prestans, mod_wsgi, netius, pycnic, Pylons, Pyramid, restlite, Tornado, Trac, TurboGears, Uliweb, uWSGI, web.py, Falcon, web2py, weblayer, Werkzeug.

Аналоги

Ruby Rack

`http://rack.github.io`

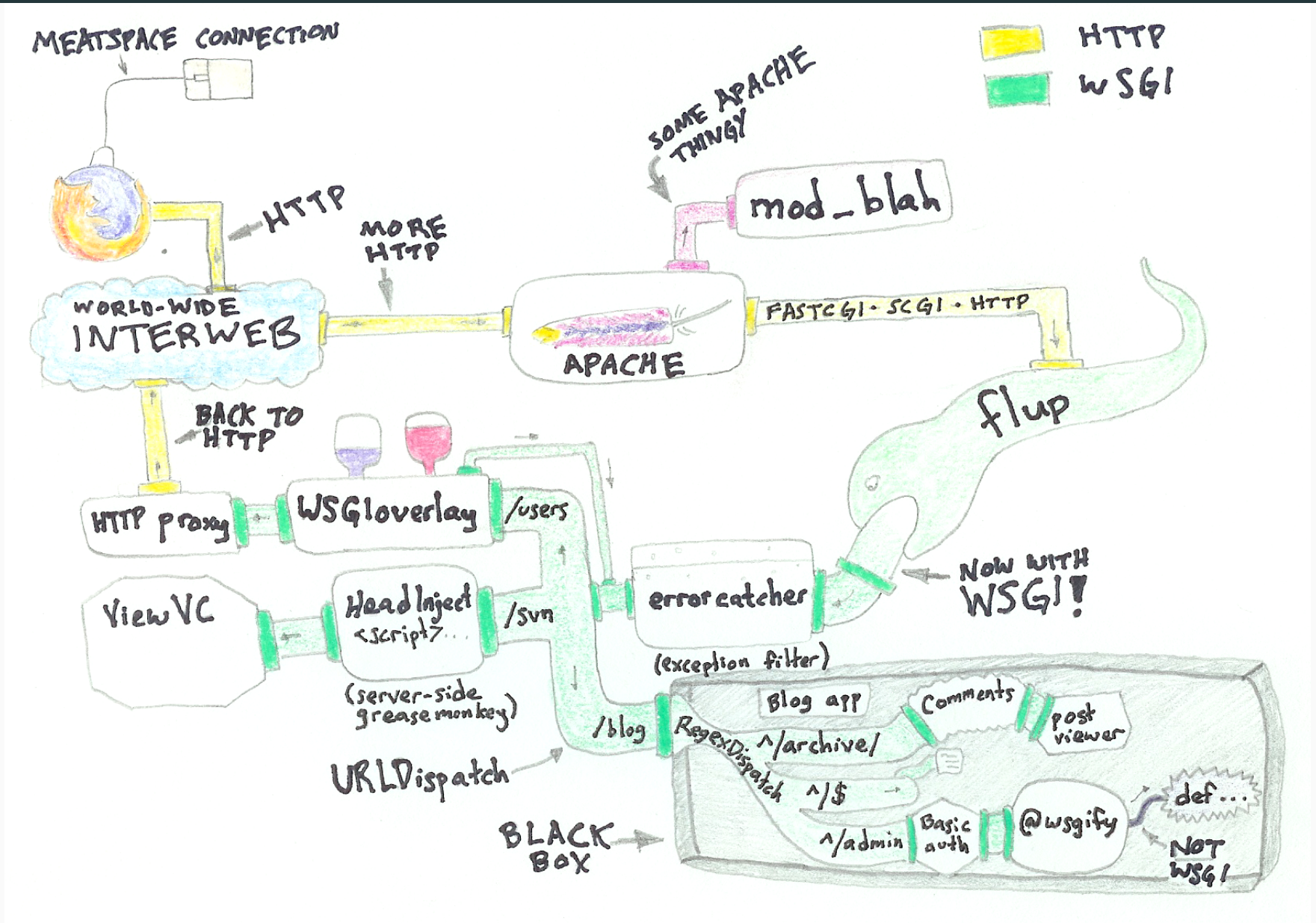
Что даёт?

- Соединять различные веб-сервера и приложения между собой, как конструктор.
- Добавлять между ними независимые программы (middleware) которые расширяют функционал.

Состоит из

- **Application**
- **Server**
- **Middleware**

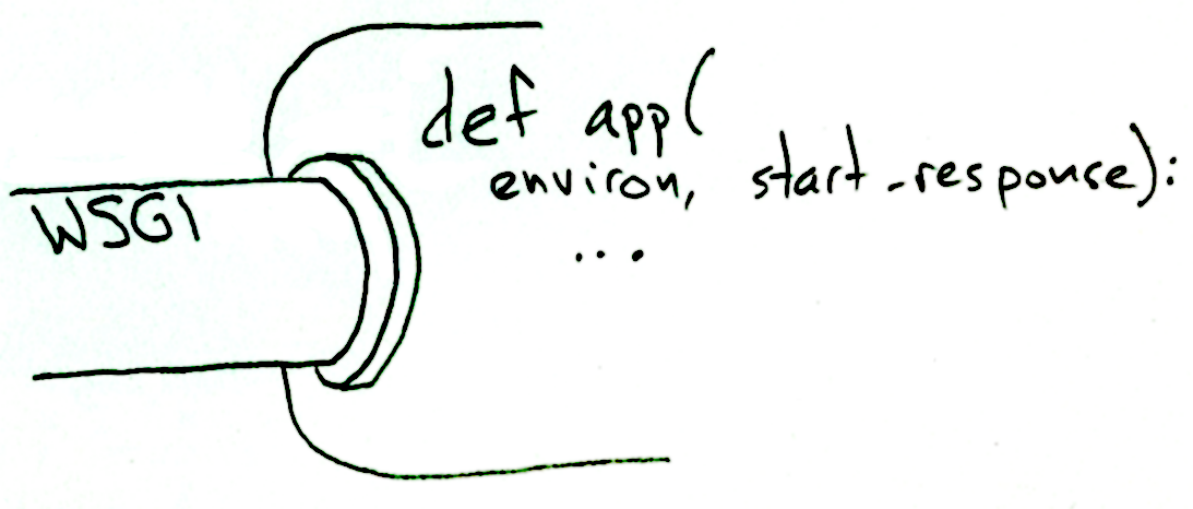
WSGI в действии



Application

- должно быть вызываемым (**callable**) объектом (обычно это функция или метод)
- принимать два параметра:
  - словарь переменных окружения (**environ**)
  - обработчик запроса (**start_response**)
- вызывать обработчик запроса с кодом HTTP-ответа и HTTP-заголовками
- возвращать итерируемый объект с телом ответа

Application



Application. Функция

Код 1: simple WSGI application

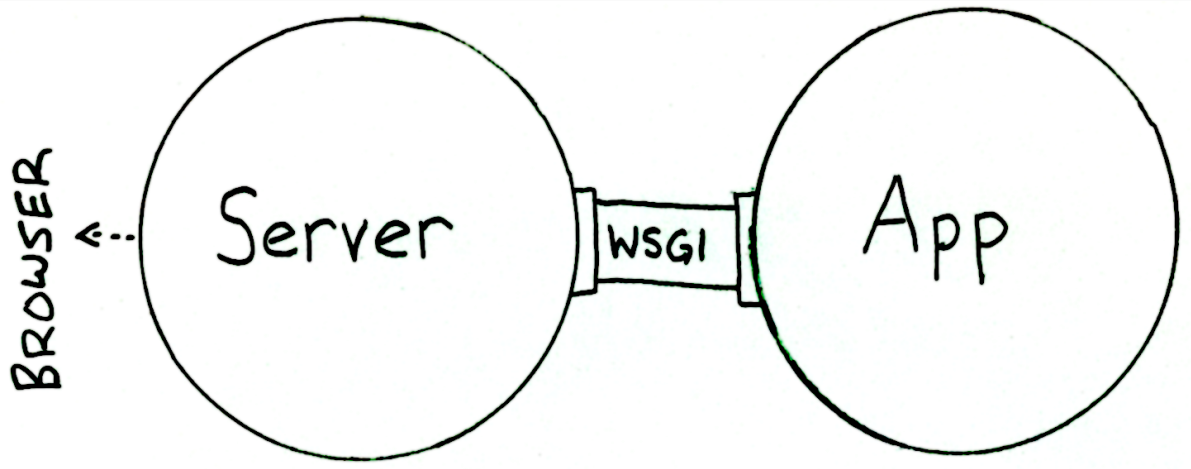
```
def simple_app(environ, start_response):
    status = '200 OK'
    response_headers = [
        ('Content-type', 'text/plain')
    ]
    start_response(status, response_headers)
    return ['Hello world!\n']
```

Application. Класс

Код 2: WSGI app class

```
class AppClass(object):
    def __init__(self, environ, start_response):
        self.environ = environ
        self.start = start_response

    def __iter__(self):
        status = '200 OK'
        response_headers = [
            ('Content-type', 'text/plain')
        ]
        self.start(status, response_headers)
        yield "Hello world!\n"
```

Server



Server. Задачи WSGI сервера

- Сформировать переменные окружения (**environment**)
- Описать функцию обработчик запроса (**start_response**)
- Передать их в **WSGI приложение**
- Результат ответа **WSGI** приложения **WSGI сервер** отправляет по HTTP, клиенту

Server. WSGI шлюз

- Либо **WSGI сервер** (шлюз) отправлет результат ответа proxy-серверу по протоколу (CGI, FastCGI, SCGI, uWSGI, …), который передает его клиенту.
- Proxy сервера могут быть например **Nginx**, **Apache**, …

Server. Что выбрать?

- Waitress
- Gunicorn
- uWSGI
- wsgiref

Server. Написать самому

```
def run_with_cgi(application):

    environ = dict(os.environ.items())
    environ['wsgi.input']        = sys.stdin
    environ['wsgi.errors']       = sys.stderr
    environ['wsgi.version']      = (1, 0)
    environ['wsgi.multithread']  = False
    environ['wsgi.multiprocess'] = True
    environ['wsgi.run_once']     = True

    if environ.get('HTTPS', 'off') in ('on', '1'):
        environ['wsgi.url_scheme'] = 'https'
    else:
        environ['wsgi.url_scheme'] = 'http'
```

Server

```
headers_set = []
headers_sent = []

def write(data):
 if not headers_set:
   raise Exception("write() before start_response")
 elif not headers_sent:
   # Before the first output, send the stored headers
   status, response_headers = headers_sent[:] \
     = headers_set
   sys.stdout.write('Status: %s\r\n' % status)
   for header in response_headers:
     sys.stdout.write('%s: %s\r\n' % header)
     sys.stdout.write('\r\n')
 sys.stdout.write(data)
 sys.stdout.flush()
```

Server

```
def start_response(status, response_headers, exc_info=N
  if exc_info:
    try:
      if headers_sent:
        # Re-raise original exception if headers sent
        raise Exception(exc_info[0], exc_info[1], exc_i
    finally:
      exc_info = None  # avoid dangling circular ref
  elif headers_set:
    raise AssertionError("Headers already set!")
  headers_set[:] = [status, response_headers]
  return write
```

Server

```
result = application(environ, start_response)
try:
  for data in result:
    if data:  # don't send headers until body appears
      write(data)
  if not headers_sent:
    write('')  # send headers now if body was empty
finally:
  if hasattr(result, 'close'):
    result.close()
```

Запуск WSGI приложения

```
run_with_cgi(simple_app)
```

```
$ python 1.cgi.app.py
Status: 200 OK
Content-type: text/plain

Hello world!
```

Middleware

То есть для сервера **middleware** является приложением, а для приложения — сервером.

Это позволяет составлять «цепочки» WSGI-совместимых **middleware**.

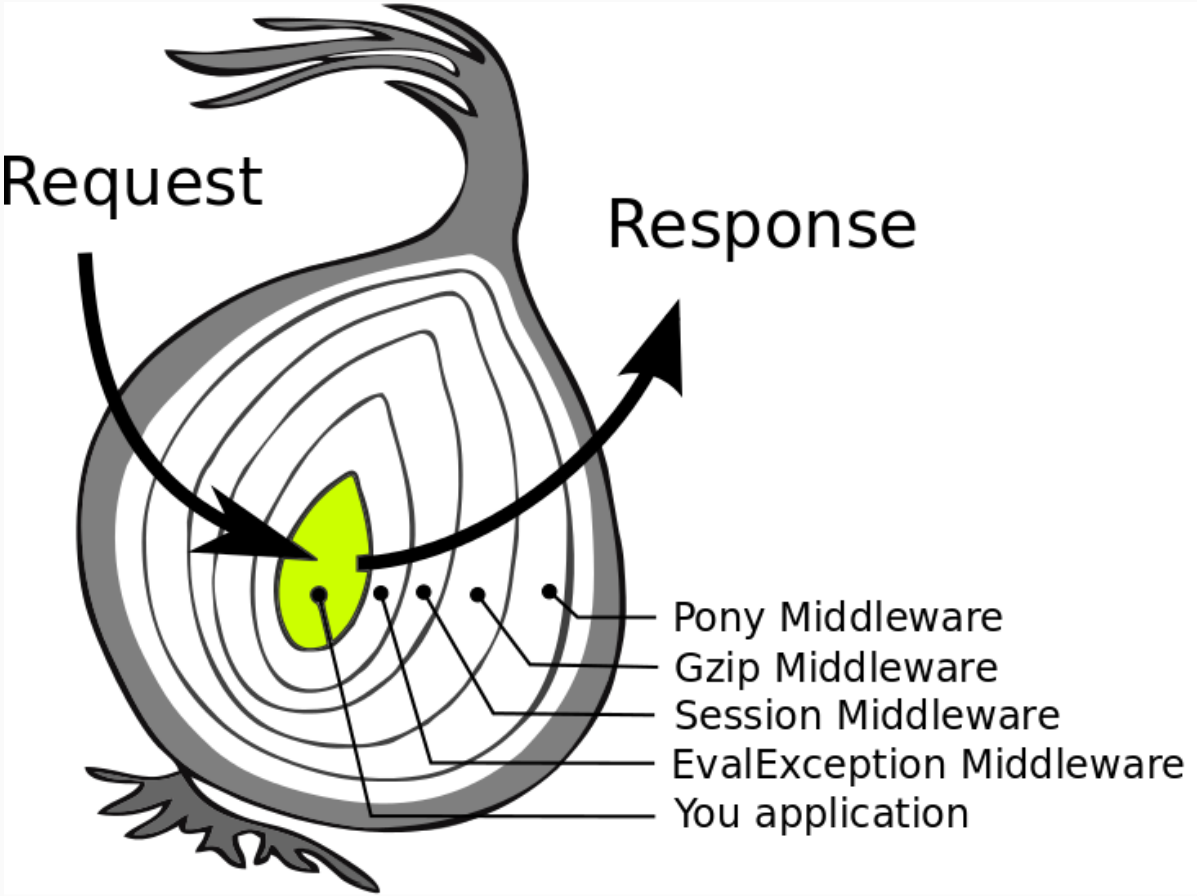
Middleware

- обработка сессий
- аутентификация/авторизация
- управление URL (маршрутизация запросов)
- балансировка нагрузки
- пост-обработка выходных данных (например, проверка на валидность)
- и прочее …

Middleware. Все вместе

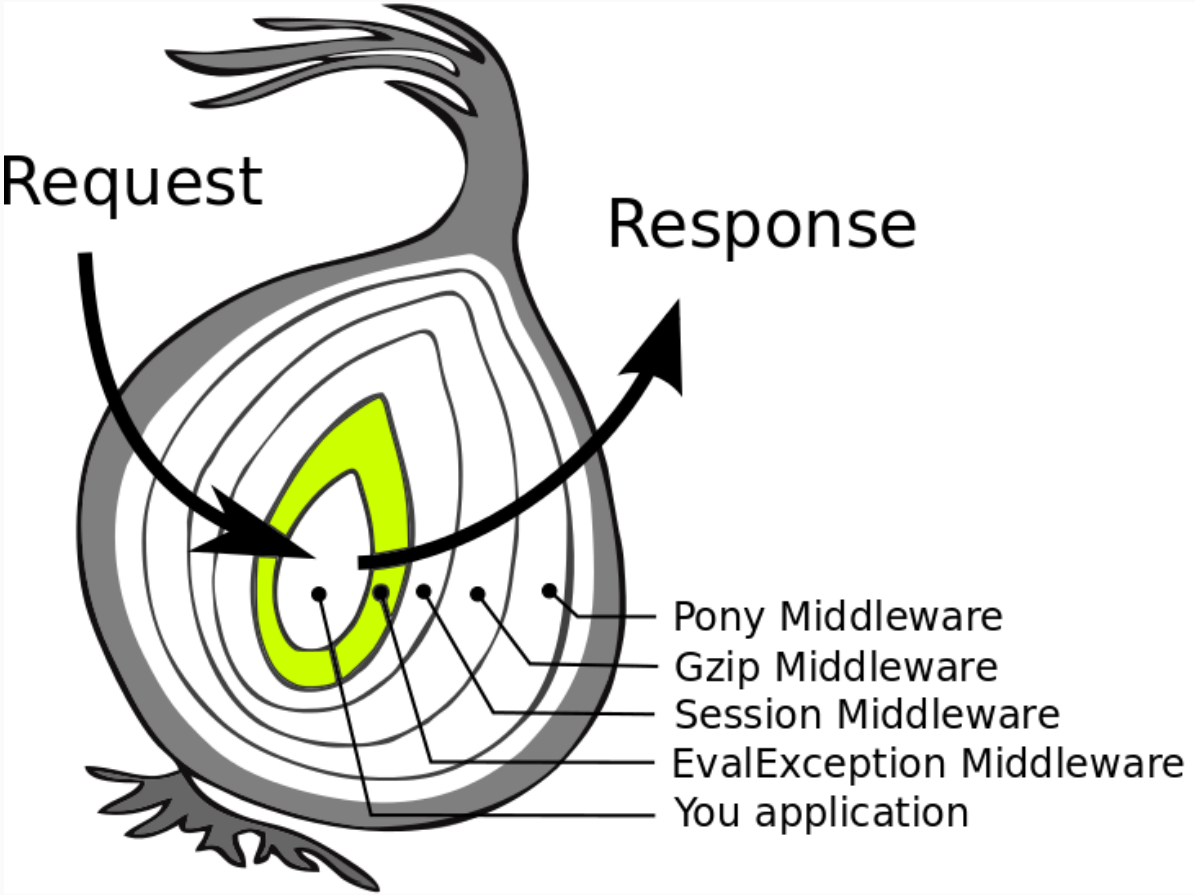
```
app = EvalException(app)   # go to /Errors
app = SessionMiddleware(app)
app = GzipMiddleware(app)
app = PonyMiddleware(app)  # go to /pony

from paste.httpserver import serve
serve(app, host='0.0.0.0', port=8000)
```

Middleware. Application

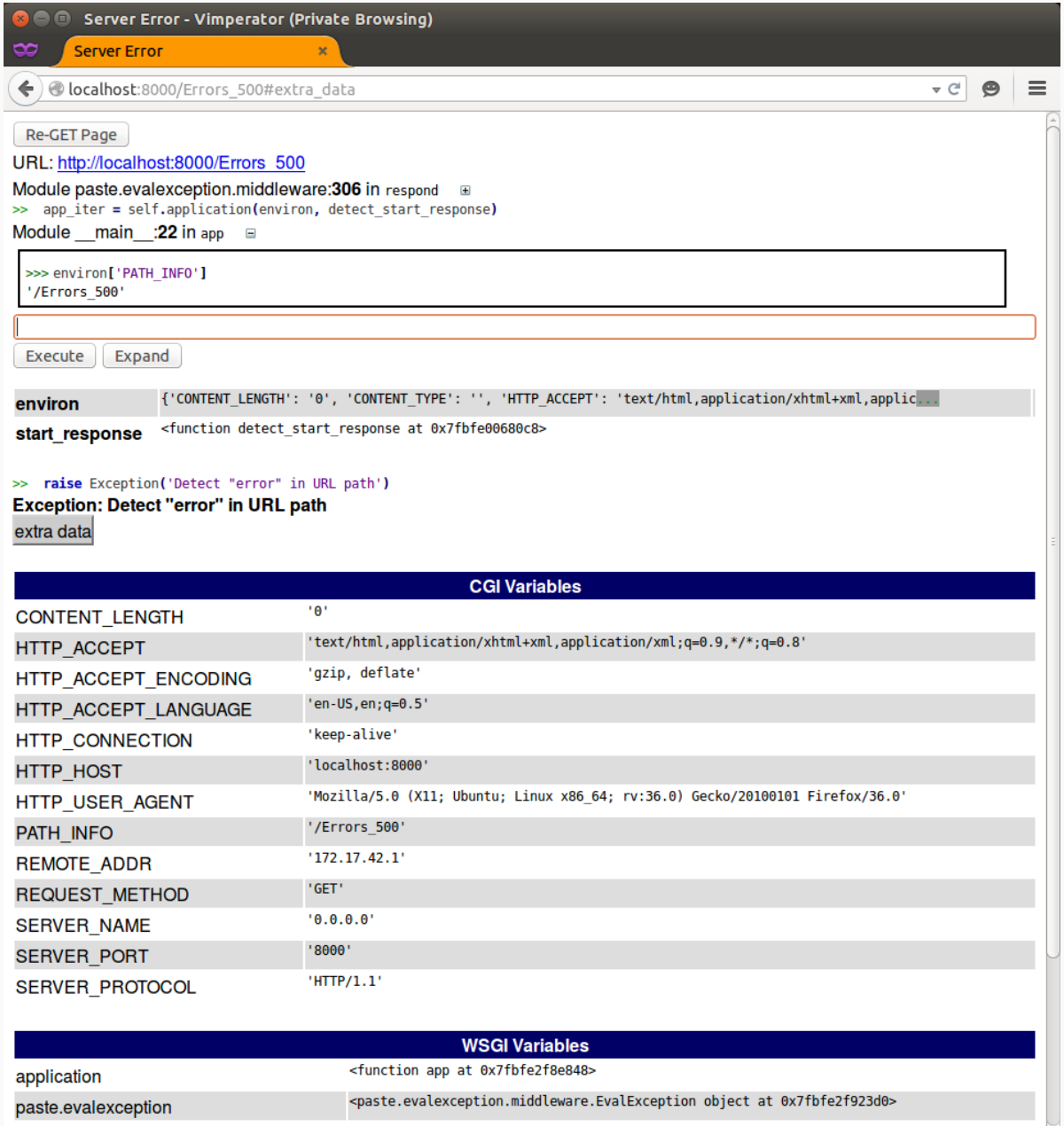


Middleware. Обработчик исключений

```
from paste.evalexception.middleware \
    import EvalException
app = EvalException(app)
```

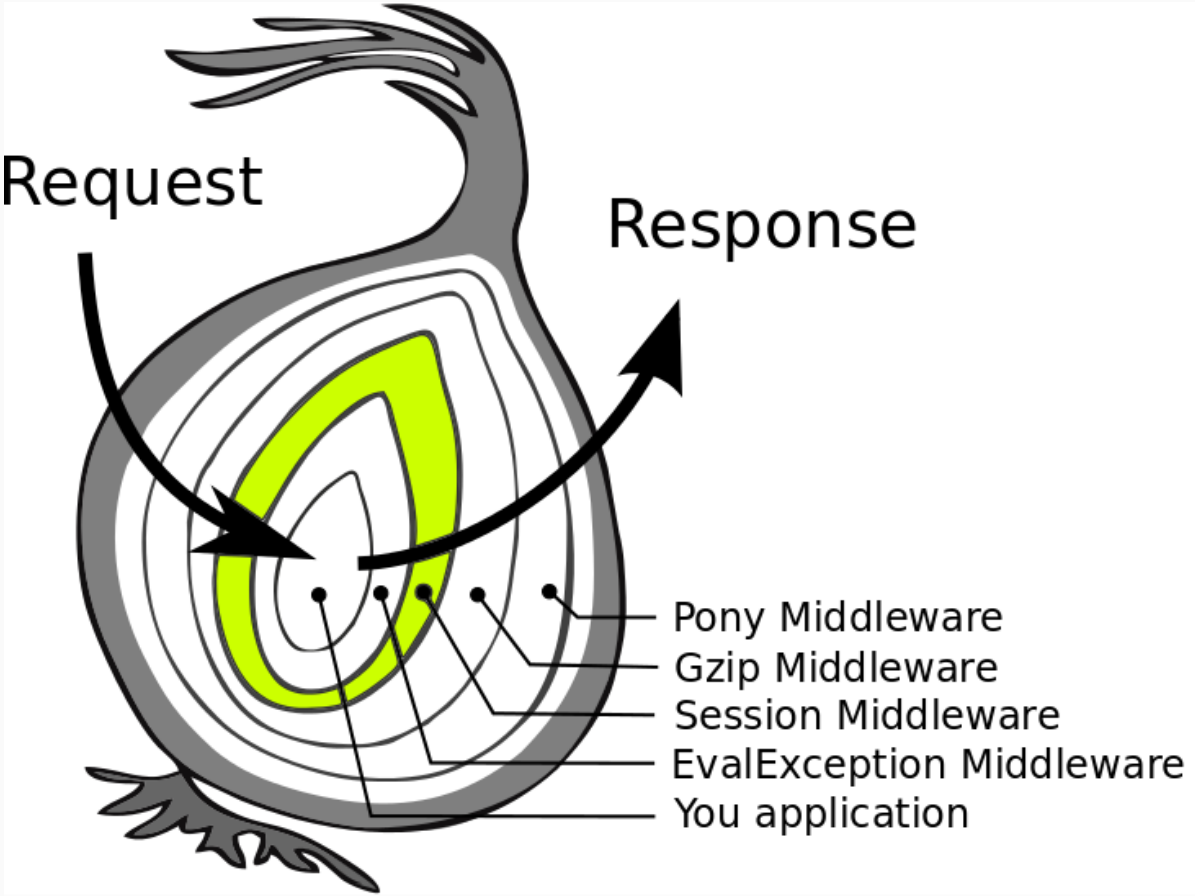


Middleware. Обработчик исключений



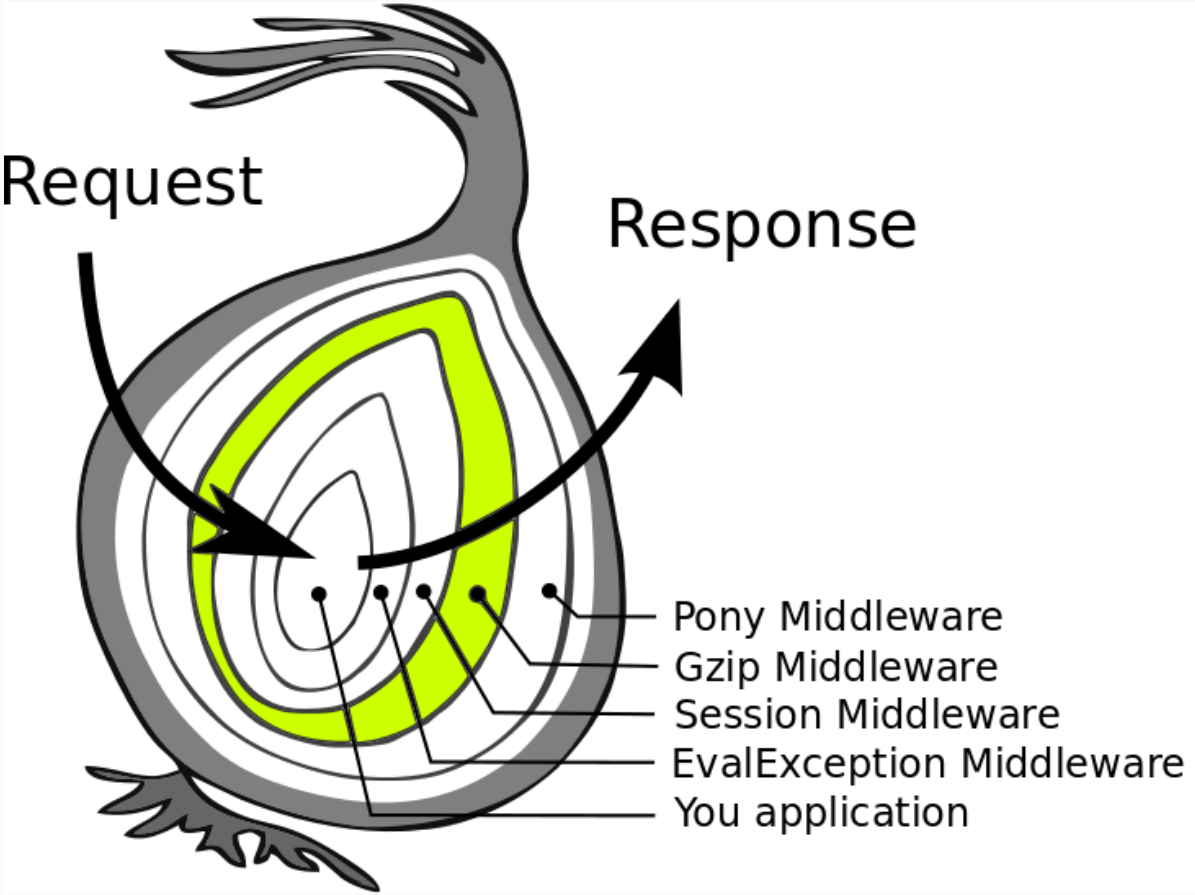
Middleware. Сессии

```
from paste.session import SessionMiddleware
app = SessionMiddleware(app)
```

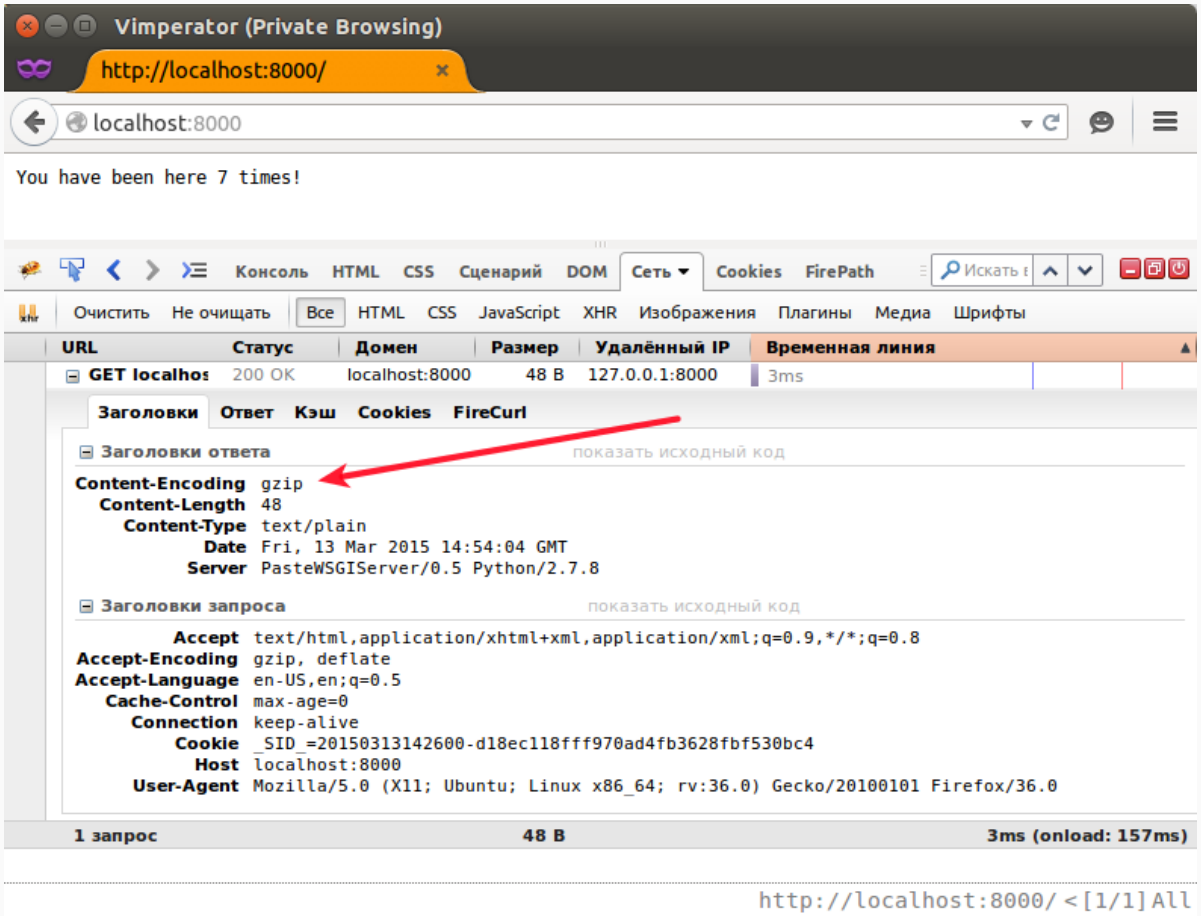


Middleware. Сжатие Gzip

```
from paste.gzipper \
     import middleware as GzipMiddleware
app = GzipMiddleware(app)
```

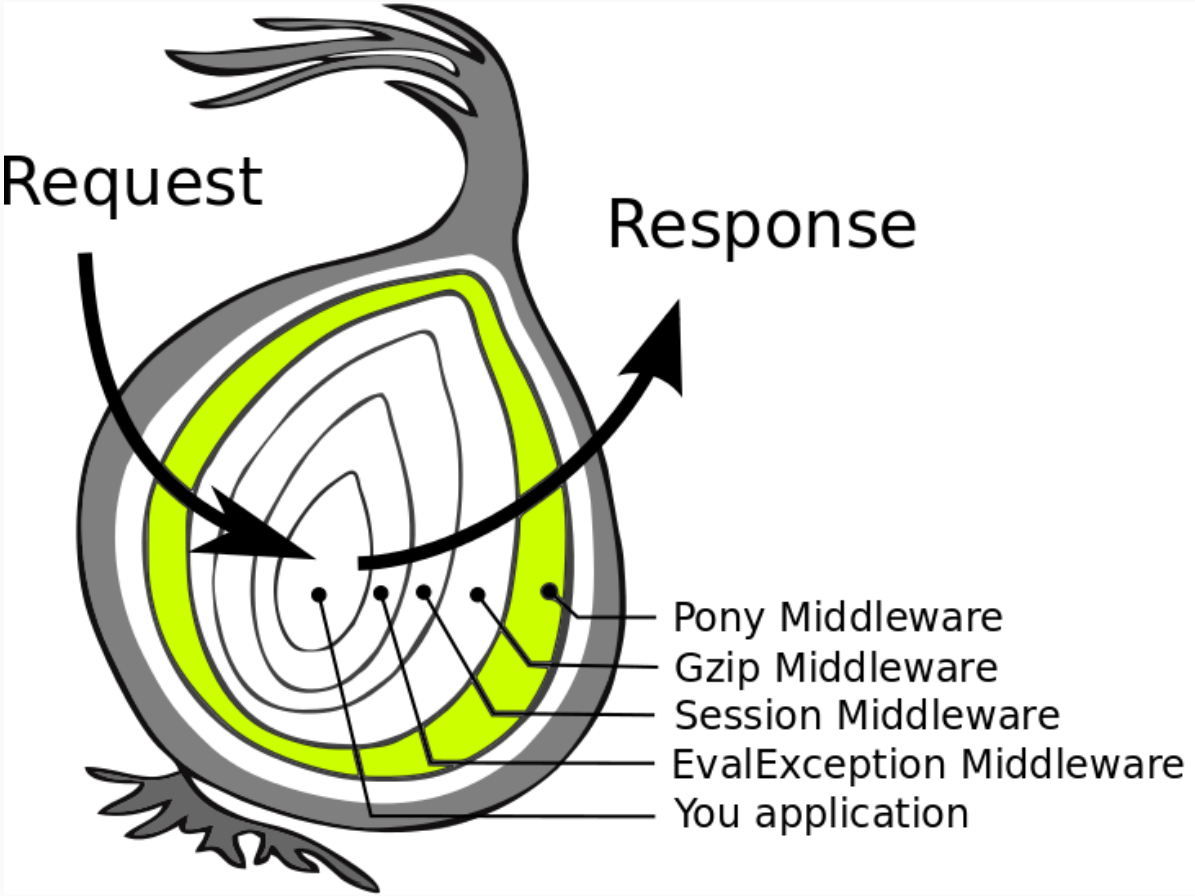


Middleware. Сжатие Gzip



Middleware. Pony

```
from paste.pony import PonyMiddleware
app = PonyMiddleware(app)
```



Middleware. Pony

Middleware. Пример

```
class GoogleRefMiddleware(object):
  def __init__(self, app):
    self.app = app

  def __call__(self, environ, start_response):
    environ['google'] = False
    if 'HTTP_REFERER' in environ:
      if environ['HTTP_REFERER']\
            .startswith('http://google.com'):
        environ['google'] = True
    return self.app(environ, start_response)

app = GoogleRefMiddleware(app)
```