



WSGI

Основы Веб-программирования

Кафедра Интеллектуальных Информационных Технологий, ИНФО, УрФУ

https:

[//en.wikipedia.org/wiki/Web_Server_Gateway_Interface](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface)

http:

[//lectureswww.readthedocs.org/5.web.server/wsgi.html](http://lectureswww.readthedocs.org/5.web.server/wsgi.html)

WSGI - это...?

Python

реп-333

реп-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.

Ruby Rack

`http://rack.github.io`

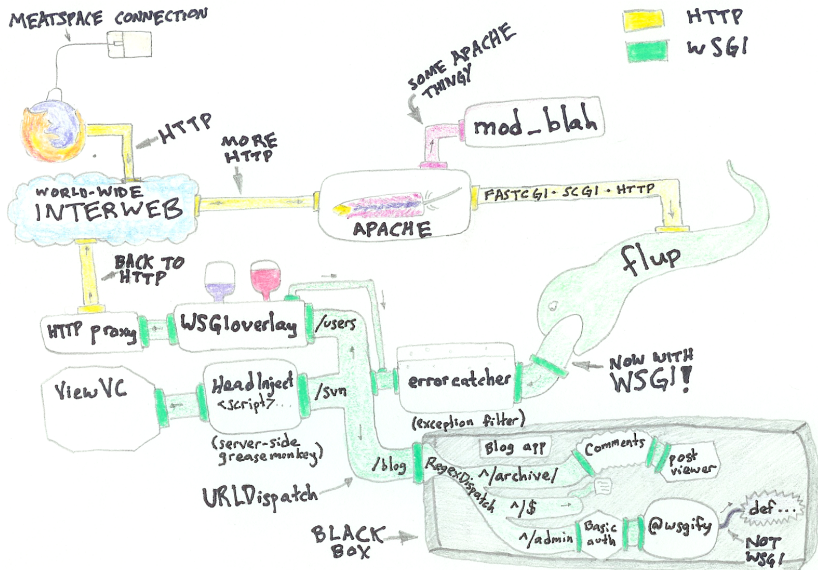
Что даёт?

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

Состоит из

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

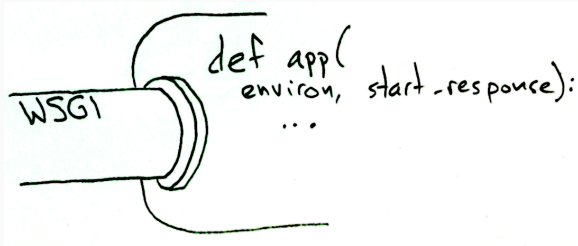
WSGI в действии



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']
```

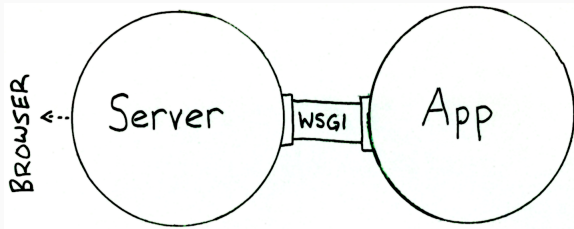
Application. Класс

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

- Либо **WSGI сервер** (шлюз) отправляет результат ответа проху-серверу по протоколу (CGI, FastCGI, SCGI, uWSGI, ...), который передает его клиенту.
- Проху 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=None):
    if exc_info:
        try:
            if headers_sent:
                # Re-raise original exception if headers sent
                raise Exception(exc_info[0], exc_info[1], exc_info[2])
        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!
```

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

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

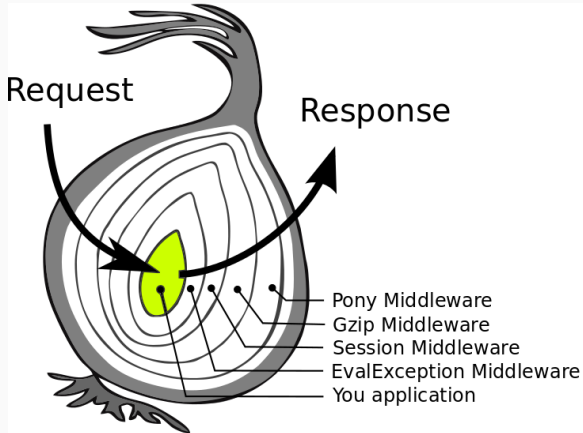
- обработка сессий
- аутентификация/авторизация
- управление URL (маршрутизация запросов)
- балансировка нагрузки
- пост-обработка выходных данных (например, проверка на валидность)
- и прочее ...

Middleware. Все вместе

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

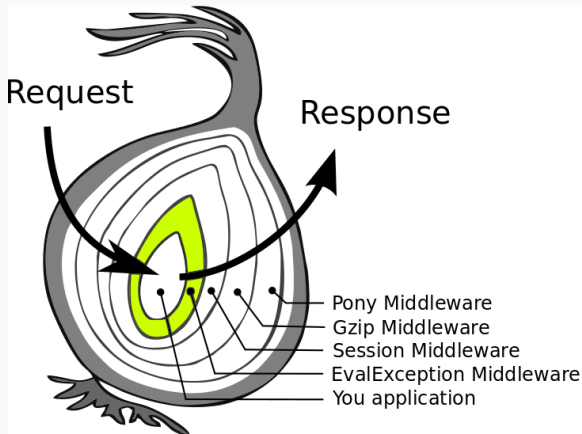
from paste.httpserver import serve
serve(app, host='0.0.0.0', port=8000)
```


Middleware. Application



Middleware. Обработчик исключений

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



Middleware. Обработчик исключений

Server Error - Vimperator (Private Browsing)

Server Error

localhost:8000/Errors_500#extra_data

Re-GET Page

URL: http://localhost:8000/Errors_500

Module `paste.evalexception.middleware:306` in respond

```
>>> app_iter = self.application(environ, detect_start_response)
```

Module `__main__:22` in app

```
>>> environ['PATH_INFO']
'/Errors_500'
```

Execute Expand

environ `{'CONTENT_LENGTH': '0', 'CONTENT_TYPE': '', 'HTTP_ACCEPT': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'}`

start_response `<function detect_start_response at 0x7bfe0600c8>`

```
>>> raise Exception('Detect "error" in URL path')
Exception: Detect "error" in URL path
```

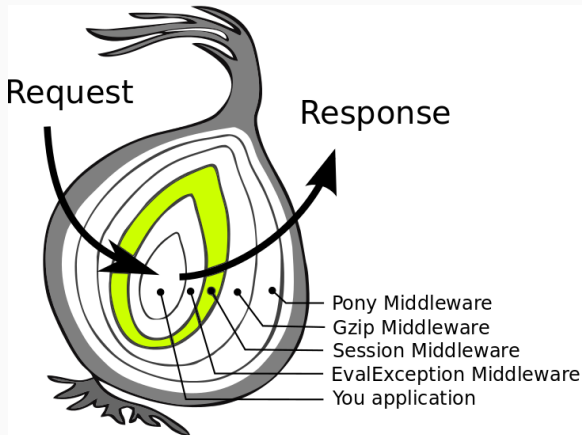
extra_data

CGI Variables	
CONTENT_LENGTH	'0'
HTTP_ACCEPT	'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8'
HTTP_ACCEPT_ENCODING	'gzip, deflate'
HTTP_ACCEPT_LANGUAGE	'en-US,en;q=0.5'
HTTP_CONNECTION	'keep-alive'
HTTP_HOST	'localhost:8000'
HTTP_USER_AGENT	'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:36.0) Gecko/20100101 Firefox/36.0'
PATH_INFO	'/Errors_500'
REMOTE_ADDR	'172.17.42.1'
REQUEST_METHOD	'GET'
SERVER_NAME	'0.0.0.0'
SERVER_PORT	'8000'
SERVER_PROTOCOL	'HTTP/1.1'

WSGI Variables	
application	<code><function app at 0x7bfe2f8e848></code>
paste.evalexception	<code><paste.evalexception.middleware.EvalException object at 0x7bfe2f923d0></code>

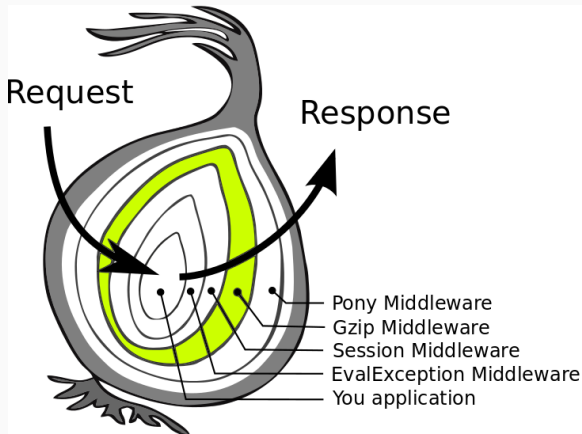
Middleware. Сессии

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



Middleware. Сжатие Gzip

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



Middleware. Сжатие Gzip

The screenshot shows a web browser window titled "Vimperator (Private Browsing)" with the address bar displaying "http://localhost:8000/". The page content says "You have been here 7 times!". Below the page, the browser's developer tools are open, showing the "Сеть" (Network) tab. A single request "GET localhost:8000" is listed with a status of "200 OK", a size of "48 B", and a response time of "3ms". The "Заголовки" (Headers) sub-tab is selected, showing the response headers. A red arrow points to the "Content-Encoding: gzip" header. The request headers also show "Accept-Encoding: gzip, deflate".

URL	Статус	Домен	Размер	Удалённый IP	Временная линия
GET localhost:8000	200 OK	localhost:8000	48 B	127.0.0.1:8000	3ms

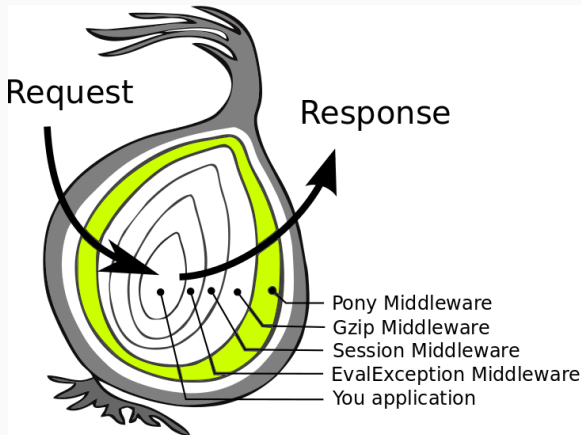
Заголовки	Ответ	Кэш	Cookies	FireCurl
Заголовки ответа <small>показать исходный код</small>				
Content-Encoding gzip				
Content-Length 48				
Content-Type text/plain				
Date Fri, 13 Mar 2015 14:54:04 GMT				
Server PasteWSGIServer/0.5 Python/2.7.8				
Заголовки запроса <small>показать исходный код</small>				
Accept text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8				
Accept-Encoding gzip, deflate				
Accept-Language en-US,en;q=0.5				
Cache-Control max-age=0				
Connection keep-alive				
Cookie _SID_=20150313142600-d18ec118ff970ad4fb3628fbf530bc4				
Host localhost:8000				
User-Agent Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:36.0) Gecko/20100101 Firefox/36.0				

1 запрос 48 B 3ms (onload: 157ms)

http://localhost:8000/ <[1/1]All

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)
```