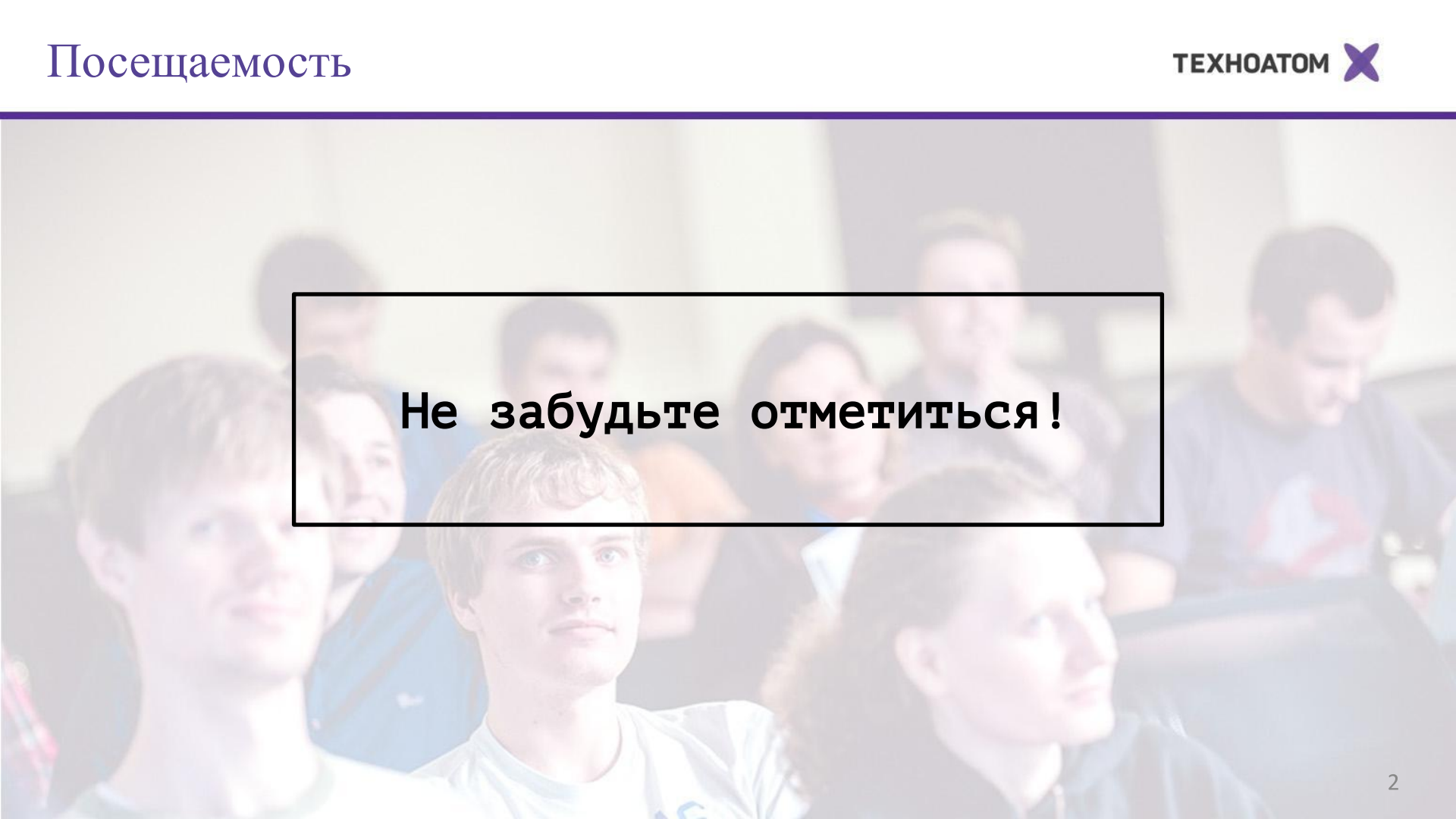


Протокол **HTTP**, **URL**, **Web**-сервер

Урок 3

Кухтичев Антон



Не забудьте отметить!

- Поговорим, что такое URL;
- Подробнее обсудим протокол HTTP;
- Расскажу, что такое веб-сервер;
- Конфигурационный файл nginx;
- Application сервер;
- WSGI протокол;
- Обсуждение квиза #1;
- Квиз #2!

Документы могут быть

- Статические
 - Это файлы на дисках сервера;
 - Как правило, обладают постоянным адресом.
- Динамические
 - Создаются на каждый запрос;
 - Содержимое зависит от времени и пользователя;
 - Адрес может быть постоянным или меняться.

URI, URL URN



- **URI** - Uniform Resource Identifier (унифицированный идентификатор ресурса);
- **URL** - Uniform Resource Locator (унифицированный локатор/указатель ресурса);
- **URN** - Uniform Resource Name (унифицированное имя ресурса).

URI является либо URL, либо URN, либо одновременно обоими.

`urn:ISBN:0-395-36341-1` - URN

<схема>:[//[<логин>[:<пароль>]@]<хост>[:<порт>]][/<URL - путь>][?<параметры>][#<якорь>]

<http://server.org:8080/path/doc.html?a=1&b=2#part1>

- http - протокол;
- server.org - DNS имя сервера (может указываться ip-адрес машины);
- 8080 - TCP порт;
- /path/doc.html - путь к файлу;
- a=1&b=2 - параметры запроса;
- part1 - якорь, положение на странице.

- `http://server.org/1.html` - абсолютный;
- `//server.org/1.html` - абсолютный (schemeless);
- `/another/page.html?a=1` - относительный (в пределах домена);
- `pictures/cat.png` - относительный (от URL текущего документа);
- `?a=1&b=2` - относительный (от URL текущего документа);
- `#part2` - относительный (в пределах текущего документа);

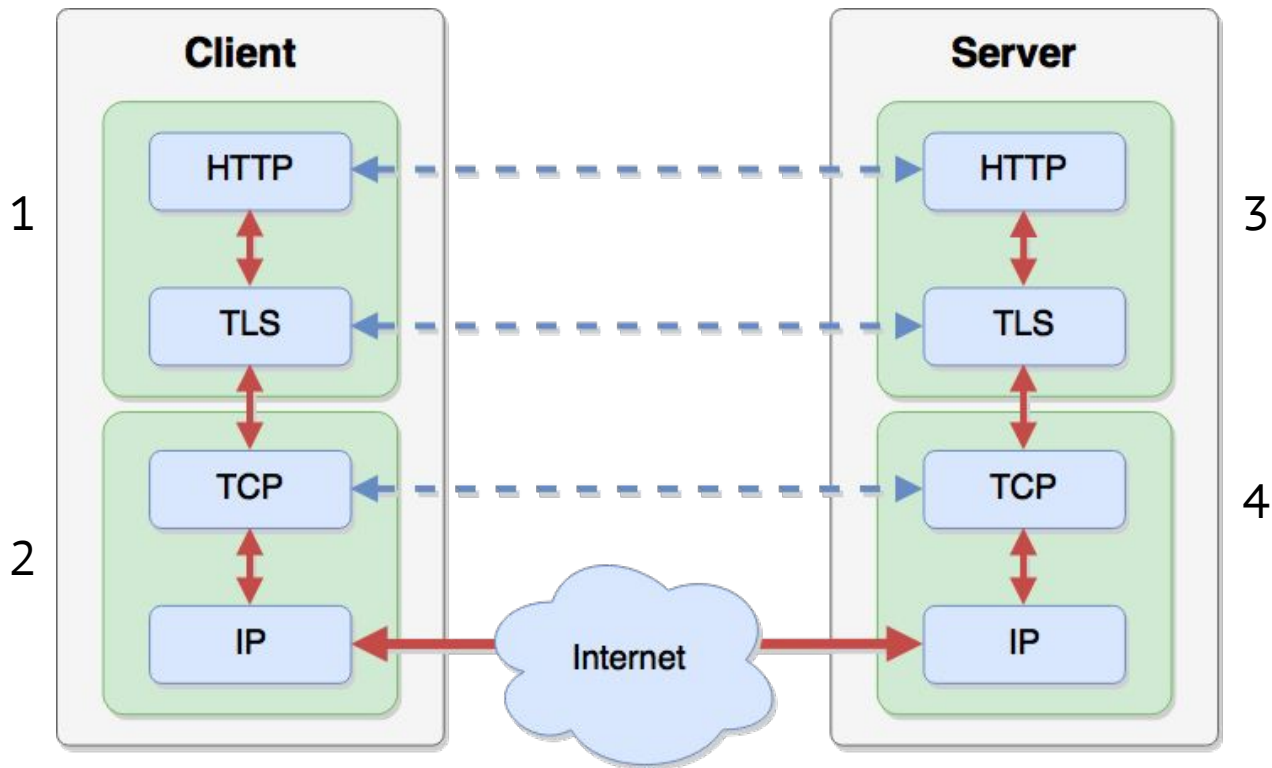
`https://site.com/path/page.html` - основной документ

- `http://wikipedia.org` = `http://wikipedia.org`
- `//cdn.org/jquery.js` = `https://cdn.org/jquery.js`
- `/admin/index.html` = <https://site.com/admin/index.html>
- `another.html` = `https://site.com/path/another.html`
- `?full=1` = `https://site.com/path/page.html?full=1`
- `#chapter2` = `https://site.com/path/page.html#chapter2`

Как происходит **HTTP** запрос

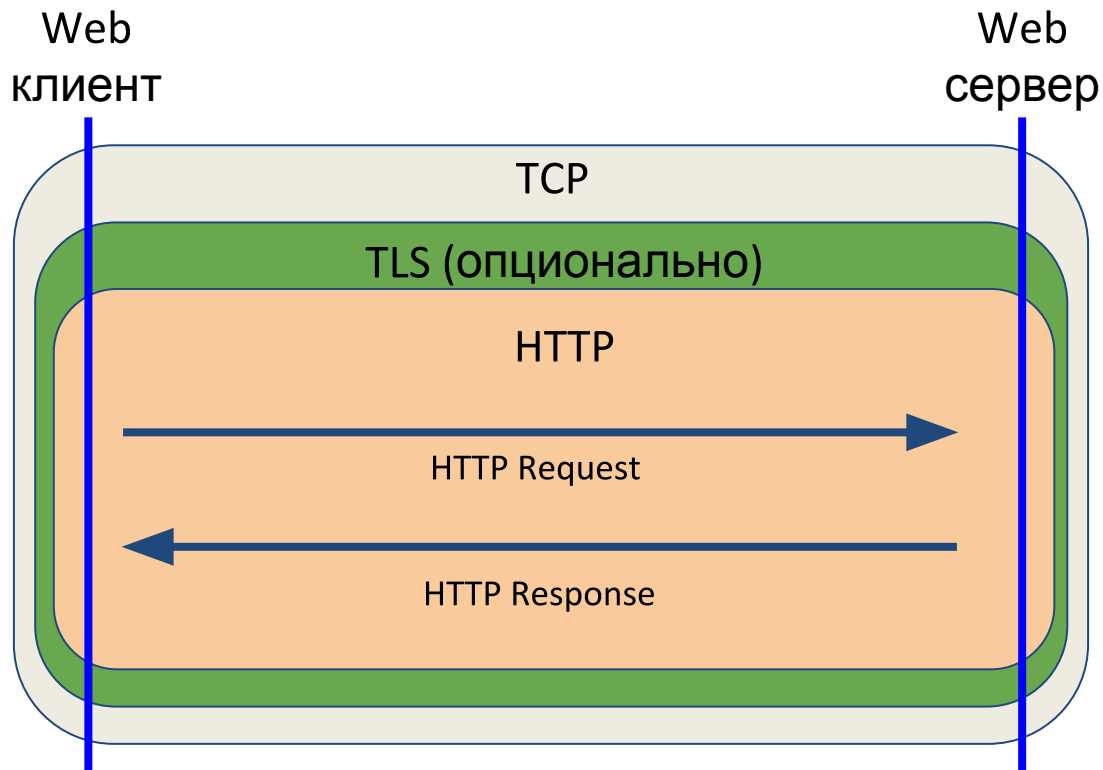
- Браузер анализирует введенный URL и извлекает имя хоста;
- Используя систему DNS, браузер преобразует домен в ip адрес;
- Устанавливает TCP соединение с web-сервером;
- Если протокол https, устанавливает TLS соединение поверх TCP;
- Формирует HTTP запрос, отправляет его, HTTP ответ;
- Браузер закрывает соединение (для HTTP/1.0);
- Далее процесс парсинга и отображения документа

Как происходит **HTTP** запрос?



HTTP протокол

- Передача документов;
- Передача мета-информации;
- Авторизация;
- Поддержка сессий;
- Кеширование документов;
- Согласование содержимого (negotiation);
- Управление соединением.



- Работает поверх TCP/TLS;
- Протокол запрос-ответ;
- Не поддерживает состояние (соединение) - **stateless**;
- **Текстовый** протокол;
- Расширяемый протокол.


```
GET http://www.ru/robots.txt HTTP/1.0
Accept: text/html, text/plain
User-Agent: telnet/hands
If-Modified-Since: Fri, 24 Jul 2015 22:53:05 GMT
```

Перевод строки - \r\n

```
GET /robots.txt HTTP/1.1
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Cache-Control: max-age=0
Connection: keep-alive
Host: www.ru
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/39.0
```

```
HTTP/1.1 404 Not Found
Server: nginx/1.5.7
Date: Sat, 25 Jul 2015 09:58:17 GMT
Content-Type: text/html; charset=iso-8859-1
Connection: close
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>...
```

- строка запроса:
 - метод
 - URL документа
 - версия
- заголовки;
- тело запроса;

- GET - получение документа;
- HEAD - получение только заголовков;
- POST - отправка данных на сервер;
- PUT - отправка документа на сервер;
- DELETE - удаление документа;
- CONNECT, TRACE, OPTIONS - используются редко;
- COPY, MOVE, MKCOL - расширения WebDAV.

- 1xx - информационные;
- 2xx - успешное выполнение;
- 3xx - перенаправления;
- 4xx - ошибка на стороне клиента;
- 5xx - ошибка на стороне сервера.

- 200 OK - запрос успешно выполнен;
- 204 No Content - запрос успешно выполнен, но документ пуст;
- 301 Moved Permanently - документ сменил URL;
- 302 Found - повторить запрос по другому URL;
- 304 Not Modified - документ не изменился, использовать кеш.

- 400 Bad Request - неправильный синтаксис запроса;
- 401 Unauthorized - требуется авторизация;
- 403 Forbidden Moved Permanently - нет доступа (неверная авторизация);
- 404 Not Found - документ не найден;
- 500 Internal Server Error - неожиданная ошибка сервера;
- 502 Bad Gateway - проксируемый отвечает с ошибкой;
- 504 Gateway Timeout - проксируемый сервер не отвечает;

Для управления соединением и форматом сообщения (документа)

- Content-Type - Mime тип документа;
- Content-Length - длина сообщения;
- Content-Encoding - кодирование документа, например, gzip-сжатие;
- Transfer-Encoding - формат передачи, например, chunked;
- Connection - управление соединением;
- Upgrade - смена протокола.

- Authorization - авторизация, чаще всего логин/пароль;
- Cookie - передача состояния (сессии) на сервер;
- Referer - URL предыдущего документа, контекст запроса;
- User-Agent - описание web-клиента, версия браузера;
- If-Modified-Since - условный GET запрос;
- Accept-* - согласование (negotiation) содержимого.

- Location - новый URL документа при перенаправлениях (коды 301, 302);
- Set-Cookie - установка состояния (сессии) в браузере;
- Last-Modified - дата последнего изменения документа;
- Date - Дата на сервере, для согласования кешей;
- Server - описание web-сервера, название и версия.

Протокол HTTP/1.0 предполагает закрытие TCP соединения сразу после ответа сервера.

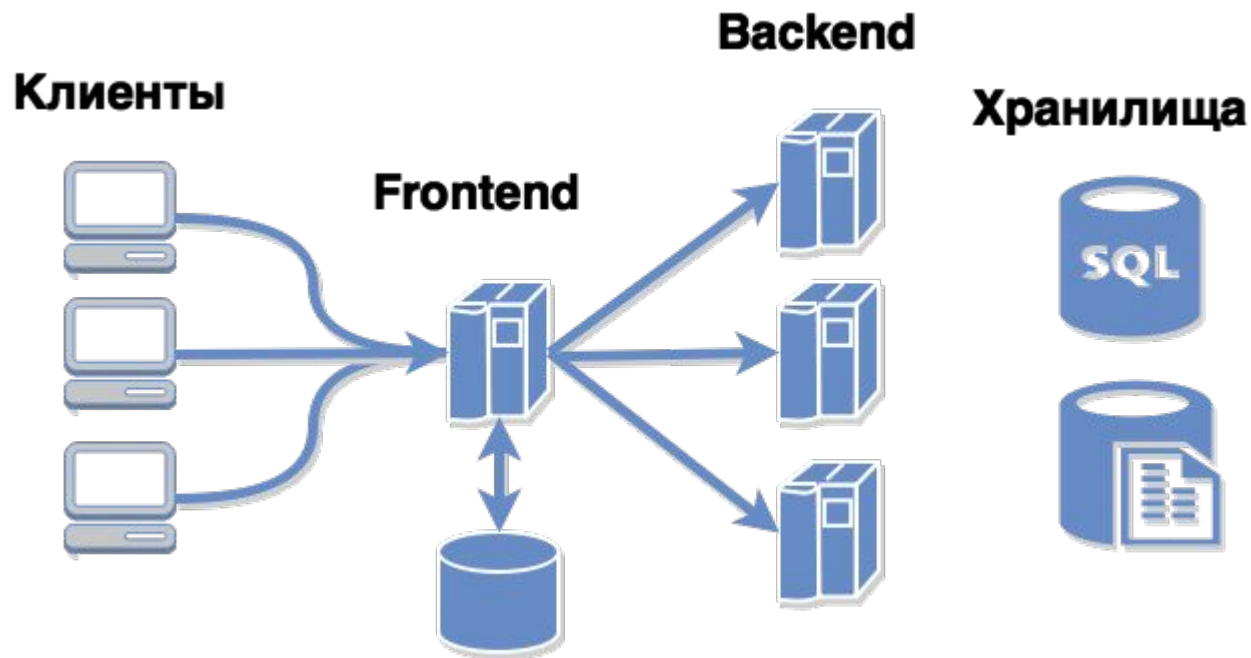
Протокол HTTP/1.1 предполагает удержание TCP соединения, если не было заголовка `Connection: close`.

Соединение должно быть закрыто, если:

- сервер или клиент использует HTTP младше 1.1;
- сервер или клиент передал заголовок `Connection: close`;
- по истечении таймаута (обычно небольшой, около 10 с);

Иначе соединение остается открытым для последующих запросов.

Трёхзвенная архитектура



- отдача статических документов;
- проксирование (reverse proxy);
- балансировка нагрузки;
- кеширование;
- сборка SSI;
- авторизация, SSL, нарезка картинок, gzip.

- frontend (медленно) читает запрос от клиента;
- frontend (быстро) передает запрос свободному backend;
- backend генерирует страницу;
- backend (быстро) возвращает ответ frontend серверу;
- frontend (медленно) возвращает ответ клиенту.

Результат: backend занят минимально возможное время.

Web сервер



Web-сервер — сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потокком или другими данными.



Microsoft
IIS

- Команда на запуск
`sudo /etc/init.d/nginx start`
- Чтение файла конфигураций;
- Получение порта 80;
- Открытие (создание) логов;
- Понижение привилегий;
- Запуск дочерних процессов/потоков;
- Готов к обработке запросов;

- Конфиг `/etc/nginx/nginx.conf`
`include /etc/nginx/sites-enabled/*`
- Init-скрипт `/etc/init.d/nginx` `[start|stop|restart]`
- PID-файл `/var/run/nginx.pid`
- Error-лог `/var/log/nginx/error.log`
- Access-лог `/var/log/nginx/access.log`

Конфигурация **Web** сервера

virtual host, вирт. хост - секция конфига web сервера, отвечающая за обслуживание определенного домена

location - секция конфига, отвечающая за обслуживание определенной группы URL


```
user    www www;
error_log /var/log/nginx.error_log info;
http {
    include      conf/mime.types;
    default_type application/octet-stream;
    log_format   simple '$remote_addr $request $status';
    server {
        listen    80;
        server_name one.example.com www.one.example.com;
        access_log /var/log/nginx.access_log simple;
        location / {
            root    /www/one.example.com;
        }
        location ~* ^.+\. (jpg|jpeg|gif)$ {
            root    /www/images;
            access_log off;
            expires 30d;
        }
    }
}
```

1. `location = /img/1.jpg`
2. `location ^~ /pic/`
3. `location ~* \.jpg$`
4. `location /img/`

При одинаковом приоритете используется тот `location`, что находите **выше** в конфиге.

- `http` – конфигурация для HTTP сервера;
- `server` – конфигурация домена (вирт. хоста);
- `server_name` – имена доменов;
- `location` – локейшен, группа URL;
- `root` , `alias` – откуда нужно брать файлы;
- `error_log` – лог ошибок сервера;
- `access_log` – лог запросов.

```
location ~* ^.+\. (jpg|jpeg|gif|png)$ {  
    root          /www/images;
```

```
}
```

```
location /sitemap/ {  
    alias /home/www/generated/;
```

```
}
```

/2015/10/ae2b5.png → /www/images/2015/10/ae2b5.png

/sitemap/index.xml → /home/www/generated/index.xml

Application

сервер

Роль application сервера заключается в исполнении бизнес-логики приложения и генерации динамических документов.

На каждый HTTP запрос application сервер запускает некоторый обработчик в приложении. Это может быть функция, класс или программа, в зависимости от технологии.

- Servlets и др. специализированные API
- mod_perl, mod_python, mod_php
- CGI
- FastCGI
- SCGI
- PSGI, **WSGI**, Rack

WSGI

WSGI, PSGI, Rack - протоколы вызова функции обработчика из application сервера. Сам application server при этом может выполняться в отдельном процессе или совпадать с web сервером. Как правило, при использовании этих протоколов в качестве application сервера выступает отдельный легковесный процесс.

```
1. pip install gunicorn
2. pip freeze > requirements.txt
3. cat myapp.py
4. def app(environ, start_response):
    data = b"Hello, world!\n"
    start_response("200 OK", [
        ("Content-Type", "text/plain"),
        ("Content-Length", str(len(data)))
    ])
    return iter([data])
4. gunicorn --workers 4 myapp:app
```

- Обработчик - функция или класс (callable);
- Метод, QueryString, заголовки запроса - через аргумент **environ**;
- Тело запроса передается через file-handle **wsgi.input**;
- HTTP код ответа и заголовки ответа передаются через вызов;
- функции **start_response**;
- Тело ответа возвращается в виде списка (*iterable*) из обработчика;
- Поток ошибок должен быть направлен в file-handle **wsgi.stderr**.

- CGI-like переменные: `REQUEST_URI` , ...
- `wsgi.version` - версия WSGI протокола
- `wsgi.url_scheme` - схема текущего URL: `https` или `http`
- `wsgi.input` - `file-handle` для чтения тела запроса
- `wsgi.errors` - `file-handle` для вывода ошибок
- `wsgi.multithreaded` - ...
- `wsgi.multiprocess` - ...

- REQUEST_METHOD - метод запроса
- PATH_INFO - путь из URL
- QUERY_STRING - фрагмент URL после ?
- REMOTE_ADDR - IP адрес пользователя
- CONTENT_LENGTH - длина тела запроса
- HTTP_COOKIE - Заголовок Cookie
- HTTP_ANY_HEADER_NAME - любой другой HTTP заголовок

- Анализ PATH_INFO и выбор конкретного обработчика;
- Разбор конкретных заголовков, например Cookie;
- Разбор QUERY_STRING;
- Разбор тела запроса:
 - x-www-form-urlencoded;
 - multipart/form-data.
- Вывод правильных заголовков ответа.

Настройка проксирования в **nginx**

```
proxy_set_header Host      $host;
proxy_set_header X-Real-IP $remote_addr;
location / {
    proxy_pass http://backend;
}
location /partner/ {
    proxy_pass http://www.partner.com;
}
location ~ /\.w\w\w?\w?$ {
    root /www/static;
}
```



```
upstream backend {  
    server back1.example.com:8080 weight=1 max_fails=3;  
    server back2.example.com:8080 weight=2;  
    server unix:/tmp/backend.sock;  
    server backup1.example.com:8080 backup;  
    server backup2.example.com:8080 backup;  
}
```

```
# wrk для Ubuntu
sudo apt-get install build-essential libssl-dev git -y
git clone https://github.com/wg/wrk.git wrk
cd wrk
make
# move the executable to somewhere in your PATH, ex:
sudo cp wrk /usr/local/bin
# ab для Ubuntu
sudo apt install apache2-utils
```

1. Установить Nginx и Gunicorn - 2 балла;
2. Настроить Nginx для отдачу статических файлов из public/ - 2 балла;
3. Создать простейшее WSGI приложение и запустить его с помощью Gunicorn - 2 балла;
4. Настроить проксирование запросов на Nginx - 2 балла;
5. Измерить производительность Nginx и Gunicorn с помощью ab или wrk - 2 балла.

Результаты квиза



Как активировать виртуальное окружение venv, созданное при помощи `python3 -m venv venv`?

1. `source venv/bin/activate`
2. `activate venv`
3. `. venv/bin/activate`
4. `import venv/bin/activate`

Как активировать виртуальное окружение venv, созданное при помощи `python3 -m venv venv`?

1. `source venv/bin/activate`
2. `activate venv`
3. `. venv/bin/activate`
4. `import venv/bin/activate`

В системе python это alias на python2.7. Создано виртуальное окружение venv при помощи python3. Активировав виртуальное окружение venv, создаём скрипт test.py (он выполняется). В скрипте есть sha-bang строка `#!/usr/bin/env python`. И вызываем скрипт `python2.7 test.py`. Какой интерпретатор запустится?

1. python3
2. python2.7
3. Результат не определён
4. Permission denied

В системе python это alias на python2.7. Создано виртуальное окружение venv при помощи python3. Активировав виртуальное окружение venv, создаём скрипт test.py (он выполняем). В скрипте есть sha-bang строка `#!/usr/bin/env python`. И вызываем скрипт `python2.7 test.py`. Какой интерпретатор запустится?

1. python3
2. python2.7
3. Результат не определён
4. Permission denied

В системе python это alias на python2.7. Создано виртуальное окружение venv при помощи python3. Активировав виртуальное окружение venv, создаём скрипт test.py (он выполняется). В скрипте есть sha-bang строка `#!/usr/bin/env python`. И вызываем скрипт `./test.py`. Какой интерпретатор запустится?

1. python3
2. python2.7
3. Результат не определён
4. Permission denied

В системе python это alias на python2.7. Создано виртуальное окружение venv при помощи python3. Активировав виртуальное окружение venv, создаём скрипт test.py (он выполняется). В скрипте есть sha-bang строка `#!/usr/bin/env python`. И вызываем скрипт `./test.py`. Какой интерпретатор запустится?

1. `python3`
2. `python2.7`
3. Результат не определён
4. `Permission denied`

Что располагается в папке `/etc/` в иерархии файловой системы Linux?

1. Файлы основных команд (утилит), которые необходимы, когда никакая другая файловая система еще не смонтирована;
2. Переменные данные;
3. Файлы конфигурации системы на данном компьютере;
4. Домашний каталог суперпользователя `root`.

Что располагается в папке `/etc/` в иерархии файловой системы Linux?

1. Файлы основных команд (утилит), которые необходимы, когда никакая другая файловая система еще не смонтирована;
2. Переменные данные;
3. Файлы конфигурации системы на данном компьютере;
4. Домашний каталог суперпользователя `root`.

Какой стек протоколов используется в протоколе HTTP?

1. HTTP-TCP-ICMP;
2. HTTP-IP-TCP;
3. HTTP-UDP-IP;
4. HTTP-TCP-IP.

Какой стек протоколов используется в протоколе HTTP?

1. HTTP-TCP-ICMP;
2. HTTP-IP-TCP;
3. HTTP-UDP-IP;
4. HTTP-TCP-IP.

В чём особенность SPA - single page application для современных web приложений?

1. Приложение располагается на одной статической html-страничке
2. html-страниц сколько угодно, но данные берутся с web-сервера
3. Каждый запрос к web-серверу сопровождается перезагрузкой страницы
4. html-контент генерируется web-сервером

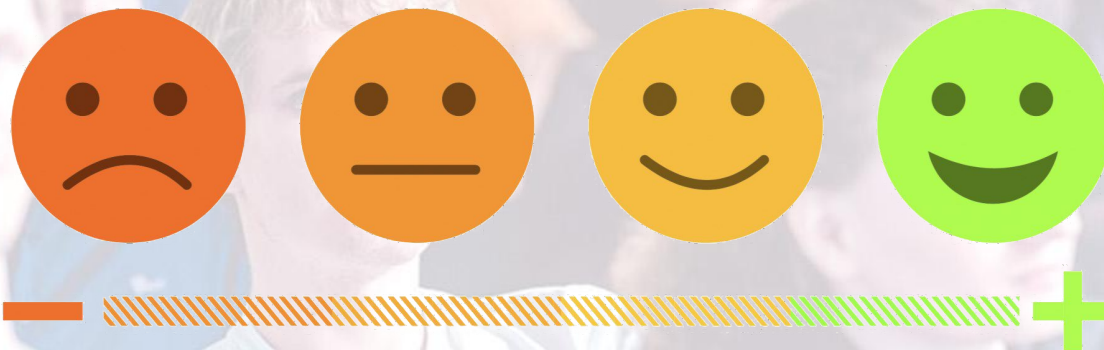
В чём особенность SPA - single page application для современных web приложений?

1. Приложение располагается на одной статической html-страничке
2. html-страниц сколько угодно, но данные берутся с web-сервера
3. Каждый запрос к web-серверу сопровождается перезагрузкой страницы
4. html-контент генерируется web-сервером

КВИЗ #2

URL: <https://forms.gle/bqv8bJXTPGcN1Nwq7>

Не забудьте оценить занятие!



Спасибо
за внимание!

Антон Кухтичев

a.kukhtichev@corp.mail.ru