

# Веб-сервер

---

## Основы Веб-программирования

Кафедра Интеллектуальных Информационных Технологий, ИНФО, УрФУ

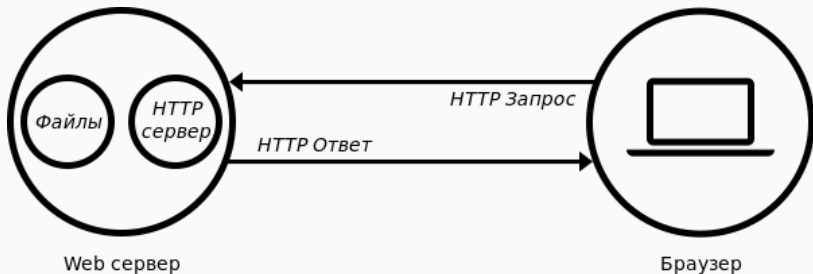
<https://ru.wikipedia.org/wiki/Веб-сервер>

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

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

# Называют

- Железо, на котором запущены сервисы и есть подключение с сети.
- Программу, которая слушает 80 порт и отвечает на HTTP запросы



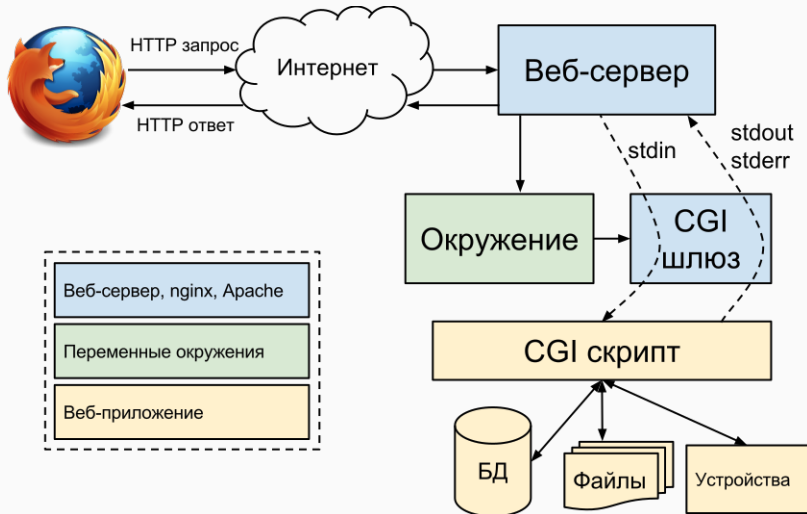
# Типы HTTP серверов

- **Статический веб-сервер**, или стек, посылает размещенные на нем файлы в браузер “как есть”. (nginx, Apache)
- **Динамических веб-сервер** состоит из статического веб-сервера плюс дополнительного программного обеспечения, наиболее часто **сервером приложений** и базы данных. (Gunicorn, Waitress, Tornado)

- Статические Веб-сайты проще всего установить. К ним относятся HTML страницы, картинки, шрифты, CSS стили и прочее...
- Динамический Веб-сайт означает, что сервер обрабатывает данные или даже генерирует их на лету из базы данных. Это обеспечивает больше гибкости, но технически сложнее в обслуживании, что делает его более сложным в разработке.

# Связь Веб-приложения с HTTP-сервером

- CGI
- FastCGI
- Встроенный сервер
- WSGI



CGI — это не язык программирования!  
Это простой протокол, позволяющий Веб-серверу передавать данные через **stdin** и читать их из **stdout**.

Поэтому, в качестве CGI-обработчика может использоваться программа на любом языке, способном работать со стандартными потоками ввода-вывода.



# CGI. Самый простой сервер.

```
python3 -m http.server -cgi 8000
```

# CGI. Hello World!

## Код 1: Си

```
#include <stdio.h>
int main(void) {
    printf("Content-Type: text/plain\n\n");
    printf("Hello, world!\n\n");
    return 0;
}
```

## Код 2: Python

```
#!/usr/bin/python
print("Content-Type: text/plain\n\nHello, world!")
```

# CGI. Переменные окружения

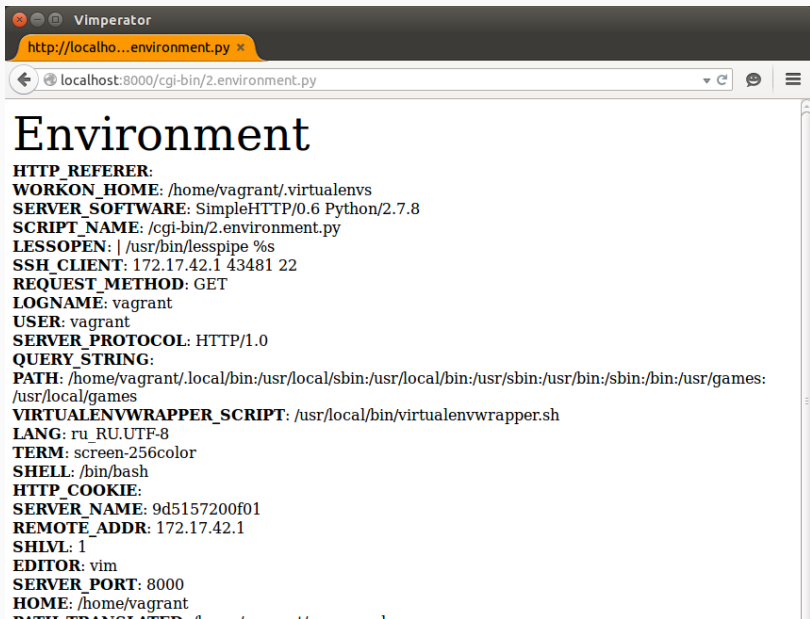
Код 3: Python

```
#!/usr/bin/python
import os

print("Content-type: text/html\r\n\r\n")
print("<font size=+10>Environment</font><br>")

for param in os.environ.keys():
    print("<b>%20s</b>: %s<br>" %
          (param, os.environ[param]))
```

# CGI. Переменные окружения



The screenshot shows a web browser window with the title "Vimperator" and the address bar containing "http://localhost:8000/cgi-bin/2.environment.py". The main content of the page is a list of environment variables and their values, displayed in a monospaced font. The variables include HTTP\_REFERER, WORKON\_HOME, SERVER\_SOFTWARE, SCRIPT\_NAME, LESSOPEN, SSH\_CLIENT, REQUEST\_METHOD, LOGNAME, USER, SERVER\_PROTOCOL, QUERY\_STRING, PATH, VIRTUALENVWRAPPER\_SCRIPT, LANG, TERM, SHELL, HTTP\_COOKIE, SERVER\_NAME, REMOTE\_ADDR, SHVL, EDITOR, SERVER\_PORT, HOME, and PATH\_TRANSLATED.

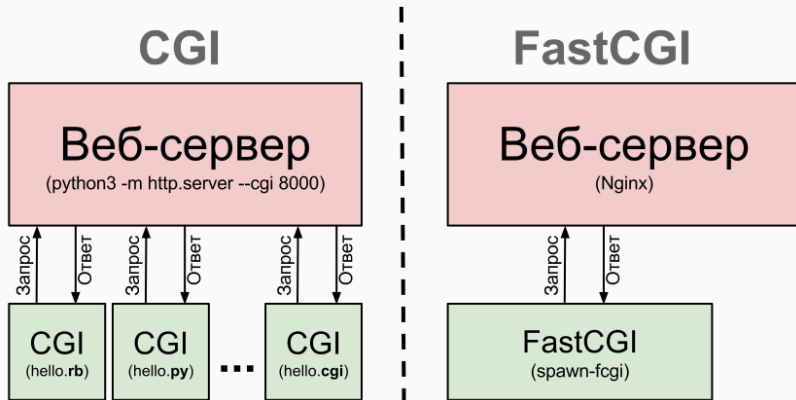
```
HTTP_REFERER:  
WORKON_HOME: /home/vagrant/.virtualenvs  
SERVER_SOFTWARE: SimpleHTTP/0.6 Python/2.7.8  
SCRIPT_NAME: /cgi-bin/2.environment.py  
LESSOPEN: | /usr/bin/lesspipe %s  
SSH_CLIENT: 172.17.42.1 43481 22  
REQUEST_METHOD: GET  
LOGNAME: vagrant  
USER: vagrant  
SERVER_PROTOCOL: HTTP/1.0  
QUERY_STRING:  
PATH: /home/vagrant/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games  
VIRTUALENVWRAPPER_SCRIPT: /usr/local/bin/virtualenvwrapper.sh  
LANG: ru_RU.UTF-8  
TERM: screen-256color  
SHELL: /bin/bash  
HTTP_COOKIE:  
SERVER_NAME: 9d5157200f01  
REMOTE_ADDR: 172.17.42.1  
SHVL: 1  
EDITOR: vim  
SERVER_PORT: 8000  
HOME: /home/vagrant  
PATH_TRANSLATED: /usr/local/bin/2.environment.py
```

## CGI. Преимущества

- Процесс CGI скрипта не зависит от Веб-сервера и в случае падения ни как не отразится на работе последнего.
- Может быть написан на любом языке программирования.
- Поддерживается большинством Веб-серверов.

## **Потребление ресурсов**

Дело в том, что каждое обращение к CGI-приложению вызывает порождение нового процесса, со всеми вытекающими отсюда накладными расходами.



- В отличие от CGI, FastCGI использует постоянно запущенные процессы для обработки множества запросов.
- FastCGI-процессы используют для связи с сервером **Unix Domain Sockets** или **TCP/IP**.



# FastCGI. Приложение

**Сборка:** gcc -o hello.fcgi hello.cpp -lfcgi

**Запуск:** spawn-fcgi -p 5000 -n hello.fcgi

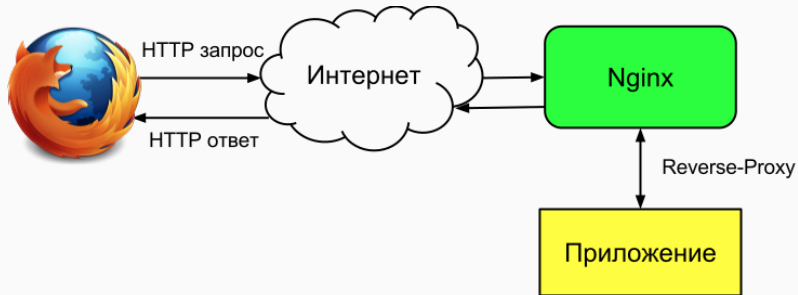
Код 4: Си

```
#include "fcgi_stdio.h"
#include <stdlib.h>

int main(void)
{
    while(FCGI_Accept() >= 0)
    {
        printf("Content-type: text/html\r\n"
              "Status: 200 OK\r\n\r\nHello World!");
    }
    return 0;
}
```

# Встроенный сервер

В некоторых языках, например Go, уже существует встроенный Веб-сервер.



# Встроенный сервер. FastCGI

В этом случае не нужно запускать отдельно fcgi сервер, например spawn-fcgi.

Код 5: Go

```
package main
import (
    "fmt"
    "net"
    "net/http"
    "net/http/fcgi"
)

func handler(res http.ResponseWriter,
             req *http.Request) {
    fmt.Fprint(res, "Hello World!")
}
```

# Встроенный сервер. FastCGI

go run hello.go

Код 6: Go

```
func main() {  
    // For local machine  
    // l, _ := net.Listen("unix",  
    //                      "/var/run/go-fcgi.sock")  
  
    // TCP 5000 listen  
    l, err := net.Listen("tcp", "0.0.0.0:5000")  
    if err != nil {  
        return  
    }  
    http.HandleFunc("/", handler)  
    fcgi.Serve(l, nil)  
}
```

# Встроенный сервер. HTTP

Может запускаться самостоятельно без дополнительных программ и настроек.

Код 7: Go

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter,
             r *http.Request) {
    fmt.Fprintln(w, "Hello World!")
}
```

# Встроенный сервер. HTTP

go run hello.go

Код 8: Go

```
func main() {  
    http.HandleFunc("/", handler)  
    http.ListenAndServe(":8000", nil)  
}
```

**WSGI** предоставляет простой и универсальный интерфейс для соединения веб-серверов, веб-приложений и прослоек между ними.

- Application
- Server
- Middleware

Все Python фреймворки поддерживают WSGI.  
Даже Django!