skypro

Основы веба. Шпаргалка

IP-адрес

> **IP-адрес** — специальный адрес, который позволяет идентифицировать компьютер или сервер в глобальной сети.

Пример IPv4-адреса: `192.168.1.1`.

Пример IPv6-адреса: `2001:0db8:85a3:0000:0000:8a2e:0370:7334`.

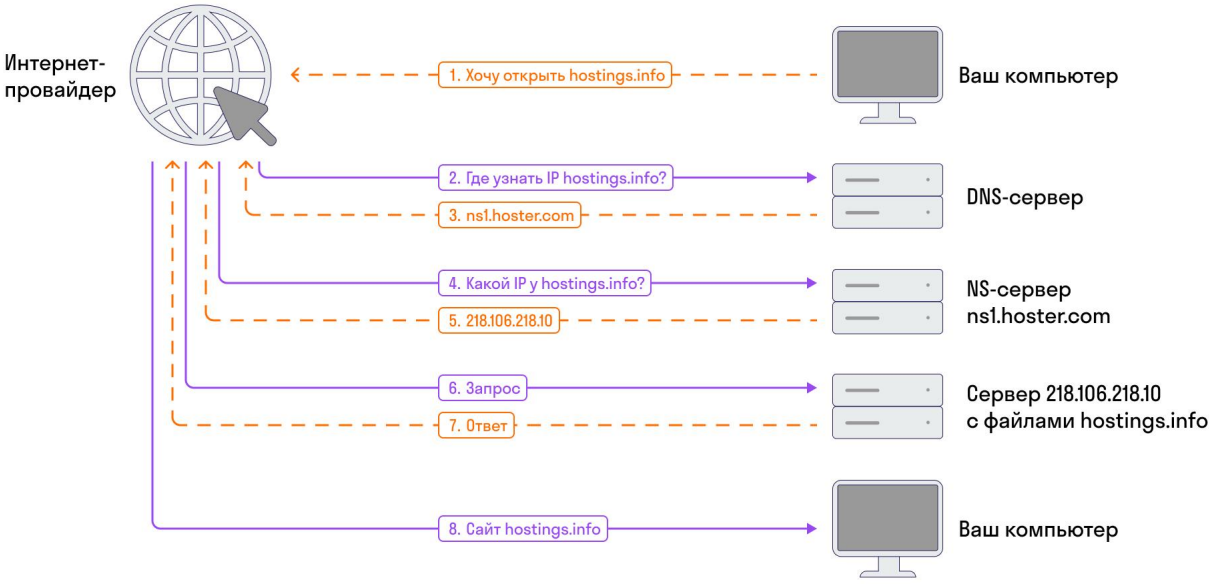
Как получить IP-адрес компьютера:

```
ip a # Linux
ipconfig # Windows
ifconfig # Unix
```

Домен и DNS

> **Домен** — уникальное имя для любого сервиса, расположенного в глобальной сети Интернет; используется для однозначного позиционирования веб-приложения и удобного обращения для пользователей.

> **DNS(Domain Name System — система доменных имен)** — это распределенная система, которая хранит и распространяет информацию о доменных именах, чаще всего используется для определения IP-адреса по имени хоста.

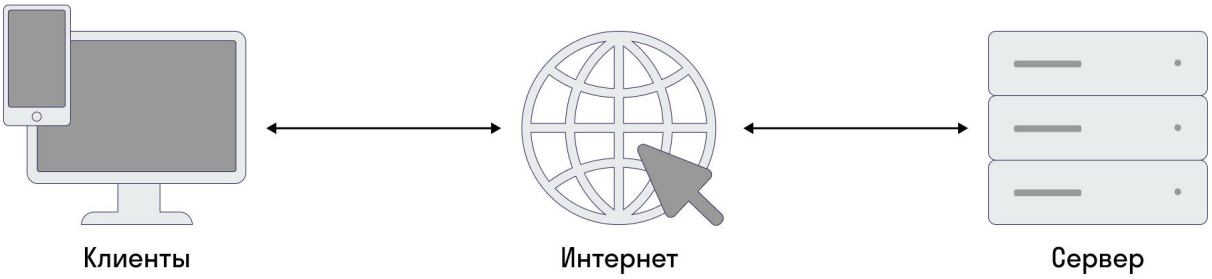


Трассировка адреса (проход всех узлов от вашего компьютера до сервера):

```
tracert google.com # Windows
traceroute google.com # Linux и macOS
```

HTTP

HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) — протокол передачи данных, который используется для передачи гипертекстовых документов в формате HTML. В настоящее время также поддерживает передачу данных в произвольном виде: JSON, XML, PDF, JPEG и т. д.



HTTP-методы

- **POST** — предназначен для направления запроса, у которого вся необходимая информация передается в теле запроса.
- **GET** — предназначен для направления запроса, который ожидает получение определенных данных на основе параметров, которые передаются в строке адреса.
- **OPTIONS** — определяет возможность сервера на обработку определенного ресурса, т. е. веб-ресурса, который может быть простым сайтом или API-сервером.
- **HEAD** — извлекает метаданные ресурса, определить доступность сервера.
- **PUT** — создает или обновляет ресурс.
- **PATCH** — частично изменяет существующий ресурс.
- **DELETE** — удаляет существующий ресурс.
- **TRACE** — расширяет ответ на запрос для понимания, какую информацию добавил каждый промежуточный сервер.

HTTP-коды ответов

Чтобы клиент мог понять, насколько успешна прошла обработка запроса, сервер возвращает ответы в виде специальных кодов, например 200, 404, 500 и так далее.

При этом эти коды разделены на группы:

- **1XX** — информационные,
- **2XX** — успешные,
- **3XX** — перенаправления,
- **4XX** — ошибки со стороны клиента,
- **5XX** — ошибки со стороны сервера.

Заголовки

На указанный URL отправляются данные с выбранным методом и добавленными заголовками:

```
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html
```

Команда curl

Отправка GET-запроса:

```
curl -H "Content-Type: application/json" -X GET http://hostname/resource
```

- Параметр `-H` — устанавливает заголовки, которые мы рассматривали выше.
- Параметр `-X` — устанавливает метод, с помощью которого будет отправлен запрос на сервер.

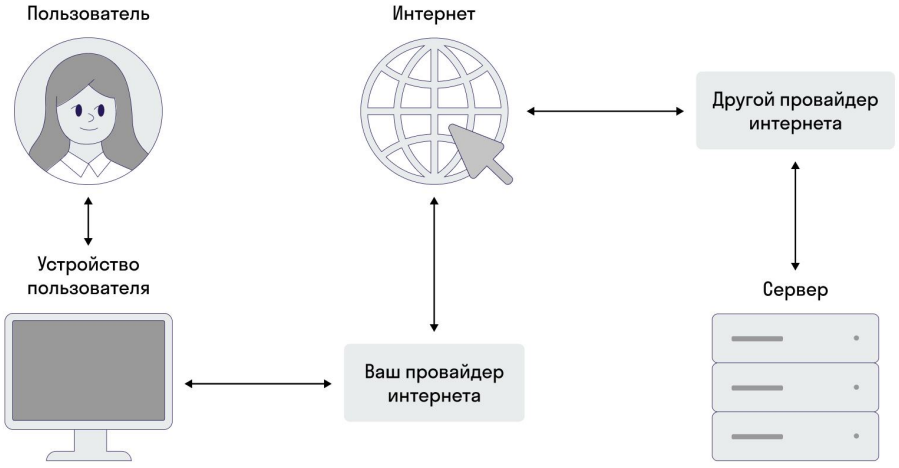
Отправка POST-запроса:

```
curl -X POST --data "param1=value1&param2=value2" http://hostname/resource
```

Параметр `--data` — задает набор параметров, которые будут переданы на сервер с POST-запросом.

Веб-приложения

> **Веб-приложение** — клиент-серверное приложение, которое использует браузер или другой инструмент, который может отправлять HTTP-запросы для обеспечения взаимодействия клиента и сервера.



Составные части веб-приложения

1. **Контроллеры** — те функции или классы, которые отвечают за обработку входящих запросов и обработку данных для вывода.

2. **Модели** — те данные для отправки пользователю по его запросу, которые хранятся в БД или другом хранилище данных.

3. **Отображения** — то, что уже видит клиент. Это может быть красивая HTML-страница с дизайном или JSON-файл, который тоже будет считаться отображением.

Пример простого веб-приложения на Python

```python
# Импорт встроенной библиотеки для работы веб-сервера
from http.server import BaseHTTPRequestHandler, HTTPServer
import time

# Для начала определим настройки запуска
hostName = "localhost" # Адрес для доступа по сети
serverPort = 8080 # Порт для доступа по сети

class MyServer(BaseHTTPRequestHandler):
    """
       Специальный класс, который отвечает за
       обработку входящих запросов от клиентов
    """
    def do_GET(self):
        """ Метод для обработки входящих GET-запросов """
        self.send_response(200) # Отправка кода ответа
        self.send_header("Content-type", "application/json")
        # Отправка типа данных, который будет передаваться
        self.end_headers() # Завершение формирования заголовков ответа
        self.wfile.write(bytes("{'message': 'OK'}", "utf-8")) # Тело ответа

if __name__ == "__main__":
    # Инициализация веб-сервера, который будет по заданным параметрах в сети
    # принимать запросы и отправлять их на обработку специальному классу,
    # который был описан выше
    webServer = HTTPServer((hostName, serverPort), MyServer)
    print("Server started http://%s:%s" % (hostName, serverPort))

    try:
        # Старт веб-сервера в бесконечном цикле прослушивания входящих запросов
        webServer.serve_forever()
    except KeyboardInterrupt:
        # Корректный способ остановить сервер в консоли через
        # сочетание клавиш Ctrl + C
        pass

    # Корректная остановка веб-сервера, чтобы он освободил адрес
    # и порт в сети, которые занимал
    webServer.server_close()
    print("Server stopped.")
```