http

珠峰培训

服务器

- 服务器可以是 **专业** 服务器也可以是 **个人** 电脑
- 能在 **特定(IP)服务器** 的 **特定端口** 上监听客户端的请求，并根据请求的路径返回相应结果都叫服务器：
- 比如霍营庆丰包子店就是一个服务器
  - 国风美唐4号楼408室(地点和门牌号)
  - 有人要吃包子可以返回包子(满足顾客的要求)

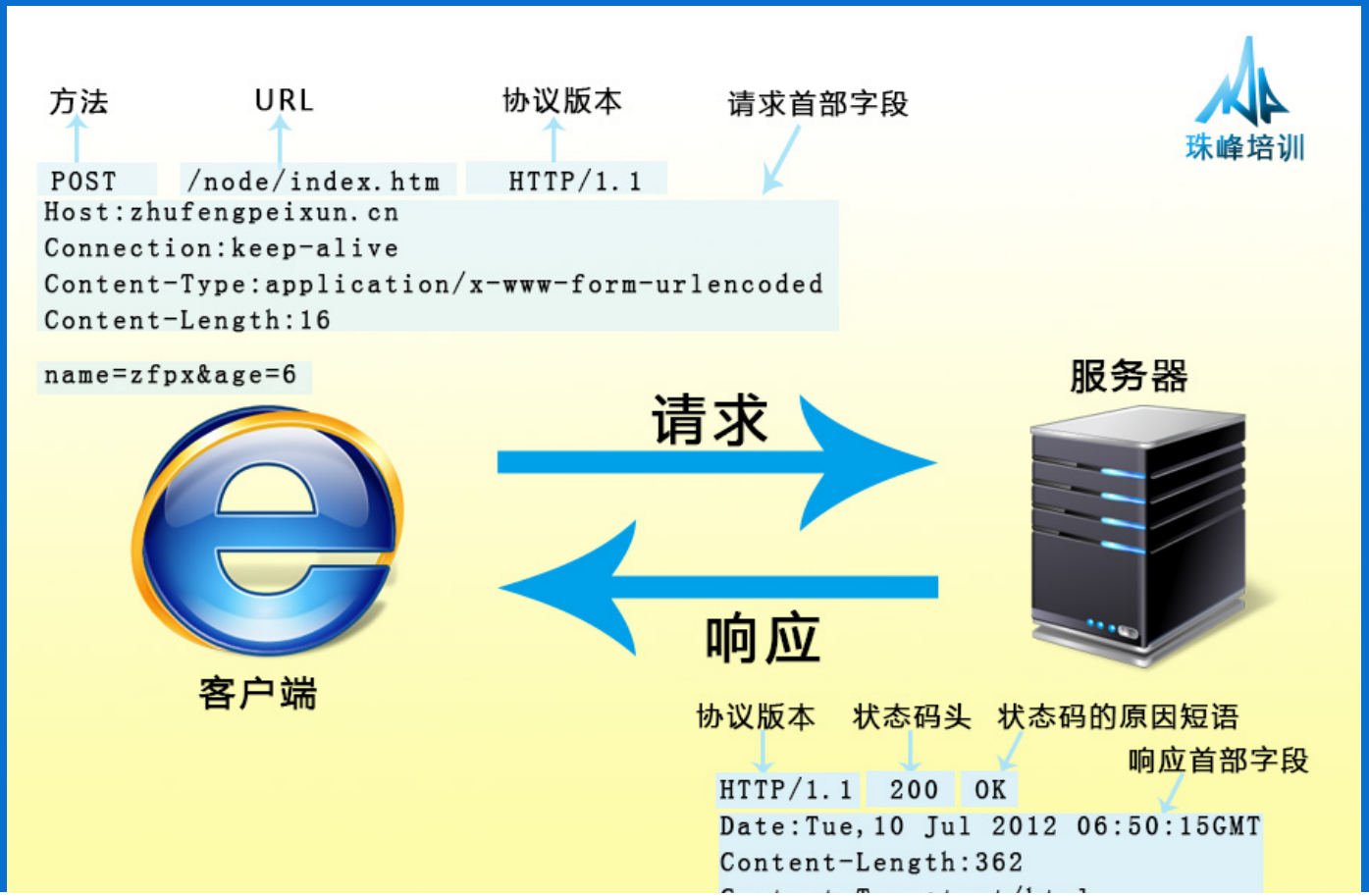
客户端

- 只要能向 **特定(IP)服务器** 的 **特定端口** 发起请求并接受响应的都叫客户端
- 可以是mac、iphone、ipad、apple等等

数据在服务器和客户端之间传递

- 可以把服务器硬盘上 **已经有的静态文件** 发送给客户端
- 也可以由服务器经过逻辑处理生成的 **动态内容** 返回给客户端，比如 **当前时间**
- 一个http事务由一条(从客户端发往服务器的)请求命令和一个(从服务器发回客户端的)响应结果组成

- 人与人之间通信，需要一种 **传输手段** (声波)和一种彼此都懂的 **语言**（比如普通话）
- 要让这些形形色色的机器能够通过网络进行交互，我们就需要指明一种 **协议** (比如 HTTP/HTTPS)和一种 **数据封装格式** (比如 HTML/JSON)
- http指的就是指的就是这种协议+数据格式的交流体系。



珠峰培训
Everest training

一个普通网站访问的过程(1)

- 浏览器(或其它客户端如微信)向服务器发出一个 **HTTP请求**
- 先把 **域名解析为IP地址** (chrome缓存1分钟(chrome://net-internals/#dns)->搜索操作系统缓存->读取本地host文件->发起DNS系统调用->运营商DNS缓存->找根域->com域)
- 客户端通过随机端口向服务器发起TCP三次握手,建立了 **TCP连接**
- 连接建立后浏览器就可以 **发送HTTP请求** 了
- 服务器接收到HTTP请求，解析请求的路径和参数，经过后台的一些处理之后 **生成完整响应** 页面
- 服务器将生成的页面作为HTTP **响应体** ，根据不同的处理结果生成 **响应头** ，发回给客户端

一个普通网站访问的过程(2)

- 客户端（浏览器）接收到 HTTP 响应,从请求中得到的 HTTP 响应体里是HTML代码，于是对HTML代码开始 **解析**
- 解析过程中遇到 **引用的服务器上的资源**（额外的 CSS、JS代码，图片、音视频，附件等），再向服务器发送请求
- 浏览器解析HTML包含的内容，用得到的 CSS 代码进行外观上的进一步 **渲染** ，JS 代码也可能会对外观进行一定的 **处理**
- 当用户与 **页面交互**（点击，悬停等等）时，JS 代码对此作出一定的反应，添加特效与动画
- 交互的过程中可能需要向服务器索取或提交额外的数据（局部的刷新）,一般不是跳转就是通过 JS 代码(响应某个动作或者定时)向服务器发送 **AJAX** 请求
- 服务器再把客户端需要的资源返回，客户端用得到的资源来实现动态效果或 **修改DOM结构** 。

- 请求的方式
  - GET 从服务器**获取资源**,比如请求一张空白的注册表单
  - POST 向服务器**提交数据** ，比如提交注册表单
- 请求的 **URL**

请求方法

- 每条http请求报文都包括一个方法表示本次将要进行何种类型的操作,如读取一个页面，删除一个资源
- **get** 系请求用来从服务器获取数据，没有请求体,不会影响服务器端的数据
- **post** 系用来将数据发送到服务器， **post** 会把要发送的数据放到请求体中，可能会影响服务器端的数据

方法	用法
GET	向服务器 **获取** 资源
POST	向服务器 **发送** 数据
DELETE	从服务器上 **删除** 资源
HEAD	仅向服务器获取 **响应头** ,不要响应体
PUT	**更新** 服务器上的一个资源

报文

请求报文

方法	用法
起始行	请求 **方法** 请求的 **URL** HTTP/ **协议版本**
请求头	**通用** 头+ **请求** 头+ **实体** 头+ **扩展** 头
请求体	发送的数据(**get** 类请求方法请求体为空)

响应报文

方法	用法
响应行	HTTP/协议 **版本** **状态码** **状态短语**
响应头	**通用** 头+ **响应** 头+ **实体** 头+ **扩展** 头
响应体	**响应** 的数据

- 状态码是一个三位数字的代码，告知响应的结果类型
- 伴随着每个数字状态码，http还会发送一条解释性的原因短语文本。

    - **状态码**
        - **1xx 请求正在处理**
        - **2xx 正常处理完成**
            - 200 OK 请求成功
        - **3xx 重定向**
            - 301 Moved Permanently 永久重定向
            - 302 Found 临时重写向
        - **4xx 客户端错误**
            - 400 Bad Request语法错误
            - 401 Unauthorized权限未认证
            - 403 Forbidden 禁止访问

MIME媒体类型

- MIME类型就是告诉浏览器用什么方式来处理这个数据。
- MIME类型是一种文本标记，表示一种主要的对象类型和一个特定的子类型，中间由一条斜杠来分隔。如 `text/html`
- MIME类型在HTTP协议中的表现为 **Request Header** 或者 **Response Header** 中的 `Content-Type`

文件类型	MIME类型
html格式的文本文档	text/html
普通的ASCII文本文档	text/plain
JPEG格式的图片	image/jpeg
GIF格式的图片	image/gif
表单	application/x-www-form-urlencoded

http模块主要用于搭建HTTP服务

- 创建HTTP服务器并 **动态响应** 当前时间

```
var http = require('http'); //表示加载http模块

// 该函数的request参数是一个对象，表示客户端的HTTP请求
// response参数也是一个对象，表示服务器端的HTTP回应。
function handle(request, response){
    //writeHead表示服务器端回应一个HTTP头信息
    response.writeHead(200, {'Content-Type': 'text/html;charset=utf-8'});
    //response.end方法表示，服务器端回应的具体内容，以及回应完成后关闭本次会话。
    response.end(new Date().toLocaleString());
}
// ceateServer方法接受一个函数作为参数
var server = http.createServer(handle);
//表示启动服务器实例，监听本机的8080端口
server.listen(8080, "127.0.0.1");
console.log('Server running on port 8080.');
```

- 将上面这几行代码保存成文件app.js，然后用node调用这个文件，服务器就开始运行了。

```
$ node app.js
```

命令行窗口将显示一行提示“Server running on port 8080.”。

读取 静态文件 并返回

```
var http = require('http');//引入http模块
var fs = require('fs');//引入读取文件的模块

http.createServer(function (request, response){
    fs.readFile('index.html', function(err, data) { //读取当前目录下面的index.html模块
        response.writeHead(200, {'Content-Type': 'text/html;charset=utf-8'}); //写响应码和响
        response.end(data); //把文件内容写入响应返回给客户端
    });
}).listen(8080, "127.0.0.1");//开始在本机的8080端口上进行监听
console.log('Server running on port 8080.');
```

根据不同的请求进行不同的响应(路由)

```
var http = require("http");

    http.createServer(function(req, res) {
        // 主页
        if (req.url == "/") {
            res.writeHead(200, { "Content-Type": "text/html;charset=utf8" });
            res.end("主页");
        }
        // 上传图片页面
        else if (req.url == "/upload") {
            res.writeHead(200, { "Content-Type": "text/html;charset=utf8" });
            res.end("上传图片");
        }
        // 404错误
        else {
            res.writeHead(404, { "Content-Type": "text/html;charset=utf8" });
            res.end("文件不存在");
        }

    }).listen(8080, "localhost");
```

处理post请求

当客户端采用`POST`方法发送数据时，服务器端可以监听`request`对象的`data`和`end`两个事件。

```
var http = require('http');

  http.createServer(function (req, res) {
      var content = "";
      console.log(req.headers.name);
      req.on('data', function (chunk) {//监听客户端的数据
          content += chunk;
      });
      req.on('end', function () {//接收完毕
          res.writeHead(200, {"Content-Type": "text/html;charset=utf-8"});
          res.write("receive: " + content);
          res.end();
      });
  }).listen(8080);
```

`data`事件会在数据接收过程中，**每收到一段数据**就触发一次，接收到

的数据被传入回调函数，`end`事件则是在**所有数据**接收完成后触发

[ajax教材] https://github.com/YataoZhang/personal_project