

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

珠峰培训就是一个服务器

回龙观东大街2楼203(地点和门牌号)

有人要听不同的课程，就提供不同的课程(满足学员们的要求)

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

数据的传递

- 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

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

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

1.http和https

1.1 http

- 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。

1.2 https

确保安全的HTTPS

HTTP+加密+认证+完整性保护=HTTPS

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法



2.网站的访问流程

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

2.1 发送请求

- 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)
- 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

2.2 得到相应

- 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

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

- 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。

3.创建服务

```
var http = require('http');
http.createServer(function (req, res) {
    res.end('你好');
}).listen(8080);
```

响应中文出现乱码问题

3.1 设置响应头

```
res.setHeader('Content-type', 'text/plain;charset=utf8');
res.statusCode = 200;
res.writeHead(200, {'Content-type': 'text/plain;charset=utf8'});
```

可以返回html文件

```
var fs = require('fs');
fs.createReadStream('./index.html').pipe(response);
```

页面中的js,css返回的均为html;需要根据路径返回不同的内容

3.2 请求信息

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

请求路径

```
console.log(req.url);
```

请求方法

```
console.log(req.method);
```

请求头信息

```
console.log(req.headers);
```

3.3 http服务

```
var http = require('http');
var fs = require('fs');
http.createServer(function (request, response) {
  if(request.url=='/hello.html'){
    response.setHeader('Content-type', 'text/html;charset=utf8');
    fs.createReadStream('./hello.html').pipe(response);
  }else if(request.url=='/favicon.ico'){
    response.statusCode = '404';
    response.end('');
  }else if(request.url=='/style.css'){
    response.setHeader('Content-type', 'text/css;charset=utf8');
    fs.createReadStream('./style.css').pipe(response);
  }else if(request.url=='/index.js'){
    response.setHeader('Content-type', 'application/x-javascript;charsets=utf8');
    fs.createReadStream('./index.js').pipe(response);
  }).listen(8080)
```

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

3.4 路径参数

当访问路径时带有查询参数,无法访问到正确路径

3.4.1 url模块

url.parse第二个参数将query变为对象格式

```
var url = require('url');  
console.log(url.parse(path,true));
```

```
Url {  
  protocol: 'http:', 协议  
  slashes: true, 是否有//  
  auth: null, 用户名密码  
  host: 'zhidao.baidu.com', 主机  
  port: null, 80 端口  
  hostname: 'zhidao.baidu.com', 主机名  
  hash: null, hash值  
  search: '?lm=0&rn=10&pn=0&fr=search&ie=gbk&word=asdasd', 路径和和  
  query: 'lm=0&rn=10&pn=0&fr=search&ie=gbk&word=asdasd', 同过true转  
  pathname: '/search', 路径  
  path: '/search?lm=0&rn=10&pn=0&fr=search&ie=gbk&word=asdasd', 路径  
  href: 'http://zhidao.baidu.com/search?lm=0&rn=10&pn=0&fr=search&i
```

根据pathname来进行路径的判断

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

3.5 改造服务

```
var urlObj = url.parse(request.url, true);
var pathname = urlObj.pathname;
if(pathname=='/'){
    response.setHeader('Content-type', 'text/html; charset=utf8');
    fs.createReadStream('./index.html').pipe(response);
}else if(pathname=='/style.css'){
    response.setHeader('Content-type', 'text/css; charset=utf8');
    fs.createReadStream('./style.css').pipe(response);
}
```

3.6 合并冗余代码

```
var http = require('http');
var fs = require('fs');
var url = require('url');
var path = require('path');
var mime = {
    '.html': 'text/html',
    '.js': 'application/x-javascript',
    '.css': 'text/css',
}
http.createServer(function (request, response) {
    var urlObj = url.parse(request.url, true);
    var pathname = urlObj.pathname;
    if(pathname=='/'){
        response.setHeader('Content-type', 'text/html; charset=utf8');
```

5. 常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1. http和https

- 1.1 http
- 1.2 https

2. 网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3. 创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4. 请求方式

- 其他常用方法


```

    fs.createReadStream('./index.html').pipe(response);
  }else if(pathname=='/favicon.ico'){
    response.statusCode = 404;
    response.end('');
  }else{
    response.setHeader('Content-type',mime[path.extname(pathname)]+'');
    fs.createReadStream('.'+pathname).pipe(response);
  }
}).listen(8080);

```

3.7 使用mime模块

3.7.1 安装mime

```
$ npm install mime
```

3.7.2 使用mime

```

var http = require('http');
var fs = require('fs');
var url = require('url');
var path = require('path');
var mime = require('mime');
http.createServer(function (request,response) {
  var urlObj = url.parse(request.url,true);
  var pathname = urlObj.pathname;

```

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

```

    if(pathname=='/'){
        response.setHeader('Content-type','text/html;charset=utf8');
        fs.createReadStream('./index.html').pipe(response);
    }else {
        var flag = fs.existsSync('./' + pathname);
        if (flag) {
            response.setHeader('Content-type', mime.lookup(pathname))
            fs.createReadStream('.') + pathname).pipe(response);
        } else {
            response.statusCode = 404;
            response.end();
        }
    }
}).listen(8080);

```

4.请求方式

发送HTTP的方法有许多种，最常用的便是 GET 和 POST ，下面就这两种进行详细地说明。

• GET

GET方法用来请求访问URI所指定的资源，（我想访问你的某个资源）并不对服务器上的内容产生任何作用结果；每次GET的内容都是相同的。GET方式把请求所需要的参数放到URL中，直接就可以在URL中看见，有大小限制。

• POST

POST方法用来传输实体主体，目的并不是获取响应的主体内容，（我要把这

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

方法信息告诉你，**用法**，POST方式则是把内容放在报文内容中，因此只要报文的内容没有限制，它的大小就没有限制。

- 总结

GET用于获取某个内容，POST用于提交某种数据请求。

按照使用场景来说，一般用户注册的内容属于私密的，这应该使用POST方式；而针对某一内容的查询，为了快速的响应，可以使用GET方式。

其他常用方法

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
OPTIONS	获取服务器上可以 支持 的方法

5.常见的状态码

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

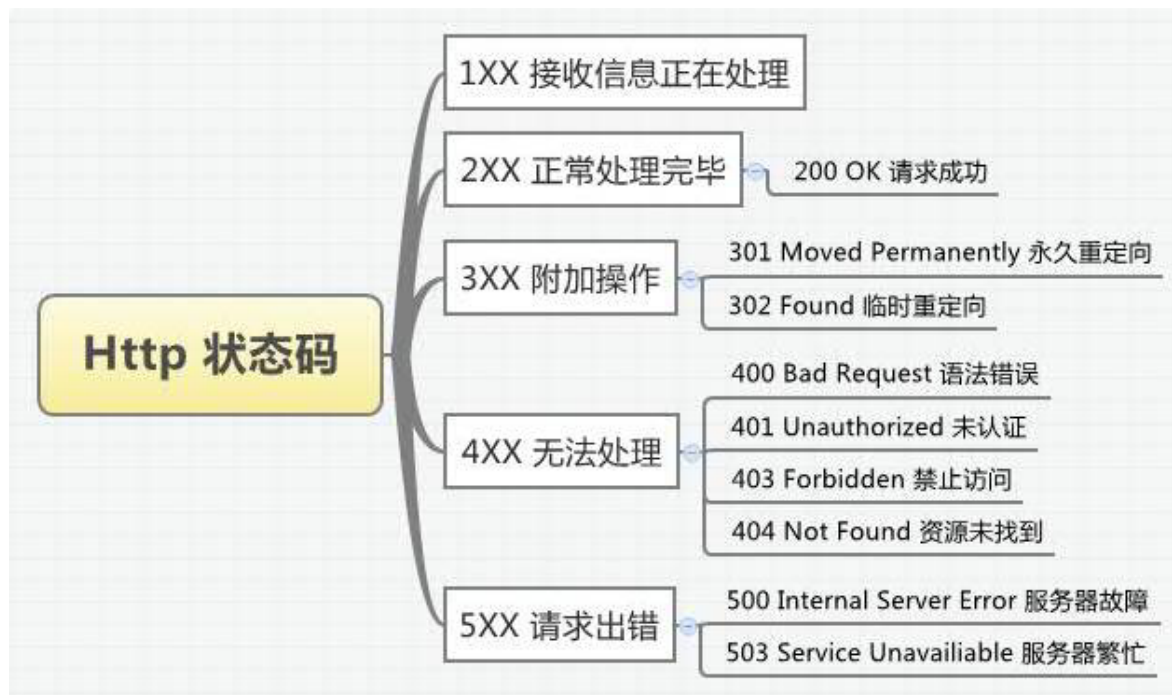
3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法



6. 查询字符串

querystring用来对查询字符串进行转换

```
var queryObj = querystring.parse(str,[sep],[eq]); //字符串转对象
var queryStr = querystring.stringify(obj,[sep],[eq]); //对象转字符串
```

- str 需要被转换的查询字符串
- sep 查询字符串中的分割字符,默认为&
- eq 查询字符串中的分配字符, 默认参数值为=

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

7. 抓取页面中的h3

```
var http = require('http');
http.createServer(function (request, response) {
  var strLar = `<!DOCTYPE html>\n
  <html lang="en">\n
  <head>\n
  <meta charset="UTF-8">\n
  <title>Title</title>\n
  </head>\n
  <body>\n
  <div></div>\n
  </body>\n
  </html>`;
  http.get('http://baijia.baidu.com', function (res) {
    var result = '';
    res.on('data', function (data) {
      result+=data;
    });
    res.on('end', function () {
      var arr = result.match(/<h3[\S\s]*?<\/h3>/g);
      var str = '<ul>';
      arr.forEach(function (item) {
        str+='<li>'+item+'</li>'
      });
      str += '</ul>';
      strLar = strLar.replace('<div></div>',str);
      response.end(strLar);
    });
  });
});
```

5.常见的状态码

6. 查询字符串

7. 抓取页面中的h3

服务器

客户端

数据的传递

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- 其他常用方法

— **מזל טוב, מזל טוב**

6. 查询字符串

7. 抓取页面中的h3

客户端

1.http和https

- 1.1 http
- 1.2 https

2.网站的访问流程

- 2.1 发送请求
- 2.2 得到相应

3.创建服务

- 3.1 设置响应头
- 3.2 请求信息
- 3.3 http服务
- 3.4 路径参数
 - 3.4.1 url模块

3.5 改造服务

3.6 合并冗余代码

3.7 使用mime模块

- 3.7.1 安装mime
- 3.7.2 使用mime

4.请求方式

- ## ■ 其他常用方法