# Ajax01-网络编程

## 1 客户端与服务器

### 1.1 定义分类

客户端：具有向服务器索取服务能力的终端，比如 手机、电脑

客户端软件：浏览器、QQ、迅雷、Foxmail等

服务器：通俗的讲，能够提供某种服务的计算机，我们称之为服务器。

按服务类型可分为：

文件服务器、数据库服务器、邮件服务器、Web 服务器等；

按操作系统可分为：

Linux服务器、Windows服务器等；

### 1.2 服务器软件

使计算机具备提供某种服务能力的应用软件，称为服务器软件

文件服务器：Server-U、FileZilla、VsFTP等；

数据库服务器：Oracle、MySQL、PostgreSQL、MSSQL、DB2等；

邮件服务器：Postfix、Sendmail等；

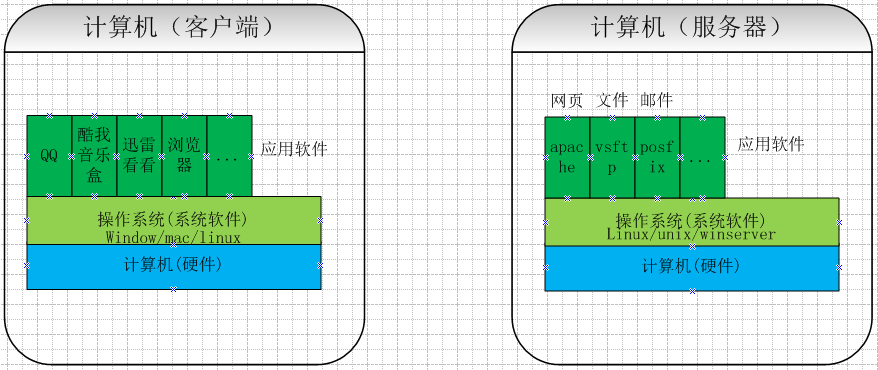
HTTP 服务器：Apache、Nginx、IIS、Tomcat、NodeJS等；

通过安装相应的服务软件，然后进行配置后就可以使计算具务了提供某种服务的能力。

### 1.3 HTTP服务器

HTTP服务器即网站服务器，主要提供文档(文本、图片、视频、音频)浏览服务。 HTTP服务器可以结合某一编程语言处理业务逻辑，由此进行的开发，通常称之为服务端开发；

常见的运行在服务端的编程语言包括 PHP、Jsp、.Net、Python、Ruby等；



### 1.4 C/S 与 B/S

S：Server

B：Browser

C：Client

C/S架构最为常见的例子是网络游戏，比如LOL，

其优点是：

1. 性能较高，可以将部分运算放在客户端，服务器只需要处理数据
2. 界面酷炫，客户端可以使用更多系统提供的效果。

其缺点是：

1. 软件迭代：需要重新下载安装
2. 不同设备访问：使用其他设备访问，需要额外安装。

B/S架构最为常见的例子是各种网站，

其优点是：

1. 更新简单：开发人员只需要更新服务器内容，用户刷新浏览器即可
2. 多设备同步：所有数据都在网上,只要能够使用浏览器即可登录使用

其缺点是：

1-性能较低：随着硬件性能的提升,这个差距在缩小；

2-兼容性问题：

### 1.5 前后端职责

既然前端是做静态效果的，那么后端是干什么的呢？

答：将前面页面的预留位置填充好数据，从新生成页面，也就是按照模板动态填充不同的内容，生成不同的页面。

以浏览器为宿主环境，结合 HTML、CSS、Javascript等技术，而进行的一系列开发，通常称之为前端开发。

从开发的角度：

1-前端的页面效果开发好后交给你后端；

2-后端去数据库获取数据，根据特定的条件填充到页面的相应位置（动态网站编程语言--- php/jsp/.net/python。。。）

3-客户端请求动态网站的php文件，php动态生成html

4-Apache返回html给浏览器

5-浏览器解析渲染页面呈现给用户

## 2 网络基础

### 2.1 IP与域名

所谓IP地址就是给每个连接在互联网上的主机分配的一个32位地址。(就像每部手机能正常通话需要一个号码一样)

查看本机IP地址 ipconfig、ifconfig

域名是一个IP地址的“面具”，由于IP地址基于数字，不方便记忆，于是便用域名来代替IP地址。

查看域名对应的IP地址 ping、tracert

### 2.2 DNS

DNS记录了 IP 地址和域名的映射（对应）关系；

查找优先级 本机hosts文件、DNS服务器。

### 2.3 端口

端口号是计算机与外界通讯交流的出口，每个端口对应不同的服务

查看端口占用情况 netstat -an

常见端口号 80、8080、3306、21、22

### 2.4 网页访问流程

访问一个网页的流程是这样的：

http://www.baidu.com

1. 通过域名查询DNS系统找到IP地址
2. 通过IP地址找到百度的服务器
3. 通过80端口找到百度服务器上的应用程序Apache或者nginx等
4. 服务器返回网页内容，
5. 浏览器把内容渲染成页面

### 2.5 网络传输协议

网络传输协议指服务器和客户端间进行通信时的约束和规范。

常见协议

1、HTTP 超文本传输协议

2、FTP 文件传输协议

3、SMTP 简单邮件传输协议等。

HTTP协议即超文本传输协议，一次HTTP连接由从客户机到服务器的请求(Request)和从服务器到客户机的响应(Response)构成。

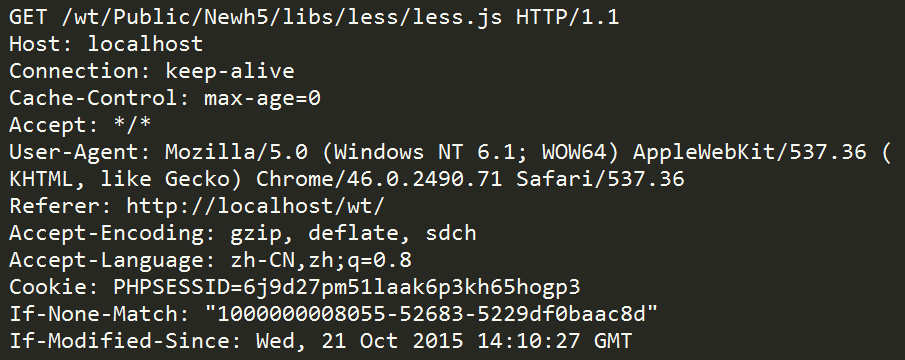


### 2.6 请求

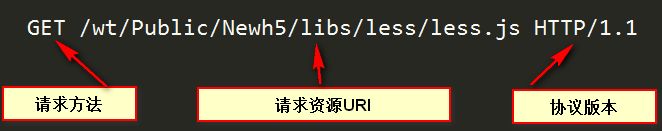
##### 2.6.1 请求组成

请求由客户端发起，格式组成：请求行、请求头、请求体

常用请求方法 POST、GET、PUT、DELETE



1)请求行



##### 2.6.2 请求头

Host：localhost请求的主机

Connection：Keep-alive 持久连接

Cache-Control：max-age=0控制缓存

Accept：\*/\* 接受的文档MIME类型

User-Agent：很重要

Referer：“我从哪里来的”

Accept-Encoding：可接受的压缩格式

If-None-Match：记录服务器响应的ETag值，用于控制缓存

If-Modified-Since：记录服务器响应的Last-Modified值

|  |
| --- |
| User-Agent：  浏览器的具体类型如：User-Agent：Mozilla/5.0 (Windows NT 6.1; rv:17.0) Gecko/20100101 Firefox/17.0  Accept：浏览器支持哪些数据类型　　如：Accept: text/html,application/xhtml+xml,application/xml;q=0.9;  Accept-Charset：浏览器采用的是哪种编码　　如：Accept-Charset: ISO-8859-1  Accept-Encoding：浏览器支持解码的数据压缩格式　　如：Accept-Encoding: gzip, deflate  Accept-Language：浏览器的语言环境　　如：Accept-Language zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3  Host：请求的主机名，允许多个域名同处一个IP地址，即虚拟主机。Host:www.baidu.com  Connection：表示是否需要持久连接。Keep-Alive/close，HTTP1.1默认是持久连接，它可以利用持久连接的优点，当页面包含多个元素时（例如Applet，图片），显著地减少下载所需要的时间。要实现这一点，Servlet需要在应答中发送一个Content-Length头，最简单的实现方法是：先把内容写入ByteArrayOutputStream，然后在正式写出内容之前计算它的大小。如：Connection: Keep-Alive  Content-Length：表示请求消息正文的长度。对于POST请求来说Content-Length必须出现。  Content-Type：WEB服务器告诉浏览器自己响应的对象的类型和字符集。例如：Content-Type: text/html; charset='gb2312'  Content-Encoding：WEB服务器表明自己使用了什么压缩方法（gzip，deflate）压缩响应中的对象。例如：Content-Encoding：gzip  Content-Language：WEB服务器告诉浏览器自己响应的对象的语言。  Cookie：最常用的请求头，浏览器每次都会将cookie发送到服务器上，允许服务器在客户端存储少量数据。  Referer：包含一个URL，用户从该URL代表的页面出发访问当前请求的页面。服务器能知道你是从哪个页面过来的。Referer: http://www.baidu.com/ |

需要注意的是,如果是往服务器提交数据,需要在请求头中设置Content-Type: application/x-www-form-urlencoded(在ajax中需要手动设置)

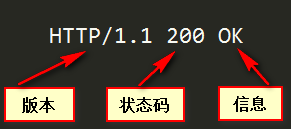
### 2.7 响应

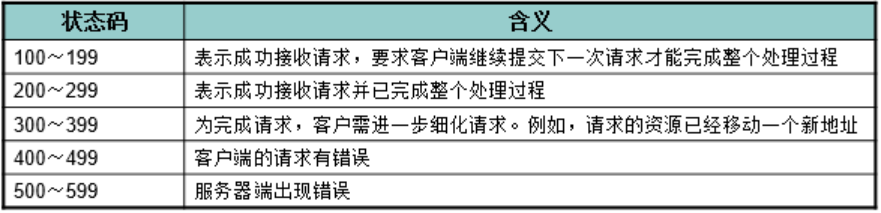
##### 2.7.1 响应组成

响应由服务器发出，格式组成：状态行、响应头、响应体



##### 2.7.2 状态行





##### 2.7.3 响应头

Date：响应时间

Server：服务器信息

Last-Modified：资源最后修改时间

ETag：资源修改后生成的唯一标识

Content-Length：响应主体长度

Keep-Alive：持久连接的时间

Content-Type：响应资源的MIME类型

|  |
| --- |
| Cache-Control  响应输出到客户端后，服务端通过该报文头属告诉客户端如何控制响应内容的缓存。  下面，的设置让客户端对响应内容缓存3600秒，也即在3600秒内，如果客户再次访问该资源，直接从客户端的缓存中返回内容给客户，不要再从服务端获取（当然，这个功能是靠客户端实现的，服务端只是通过这个属性提示客户端“应该这么做”，做不做，还是决定于客户端，如果是自己宣称支持HTTP的客户端，则就应该这样实现）。  Cache-Control: max-age=3600  ETag  一个代表响应服务端资源（如页面）版本的报文头属性，如果某个服务端资源发生变化了，这个ETag就会相应发生变化。它是Cache-Control的有益补充，可以让客户端“更智能”地处理什么时候要从服务端取资源，什么时候可以直接从缓存中返回响应。  ETag: "737060cd8c284d8af7ad3082f209582d"  Location  我们在Asp.net中让页面Redirect到一个某个A页面中，其实是让客户端再发一个请求到A页面，这个需要Redirect到的A页面的URL，其实就是通过响应报文头的Location属性告知客户端的，如下的报文头属性，将使客户端redirect到iteye的首页中：  Location: http://www.google.com.hk  Set-Cookie  服务端可以设置客户端的Cookie，其原理就是通过这个响应报文头属性实现的。  Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1  HTTP响应体：如果请求的是HTML页面，那么返回的就是HTML代码。如果是JS就是JS代码。  HTTP响应头：而设置Cookie，缓存等信息就是在响应头属性设置的。  HTTP响应行：主要是设置响应状态等信息。 |

### 2.8 协商缓存

利用浏览器的缓存机制，可以有效的减少HTTP的请求，提高页面加载速度，增强用户体验，同时也能极大的减轻服务器的负担，结合HTTP协议，缓存协商就是根据HTTP协议实现缓存控制的一种机制。

## 3 WAMP集成环境

### 3.1 安装WAMP

WAMP:Windows + Apache + Mysql + PHP，首字母组合。

安装wampserver，和普通软件安装无差别，除指定安装路径外，其它默认安装，任务图标绿色为正常启动状态。

XAMP：XAMPP可以安装在Linux、macOS等系统上。

127.0.0.1是回送地址，指本机。

WAMP显示黄色解决方案：

1. 服务没有启动完全：左键点击图标，启动所有服务；
2. 微软IIS服务问题：打开控制面板，点击程序-启用或关闭windows功能-保证最后的Internet Information Services 没有被选中。
3. 修改端口号：在httpd.conf中搜素listen，修改为8080，重启wamp，访问地址君加入8080即可。

### 3.2 配置网站根目录

网站根目录是Web服务器上存放网站程序的空间。

1-wampserver安装目录下bin\apache\Apache2.2.21\conf\httpd.conf

2-查找documentRoot并修改



例如：这样就指定了 "D:/www/"为存放网站的根目录。



3-配置根目录，查找



修改成



4-修改完后，并不能立即生效，需要重启Apache

### 3.3 配置虚拟机

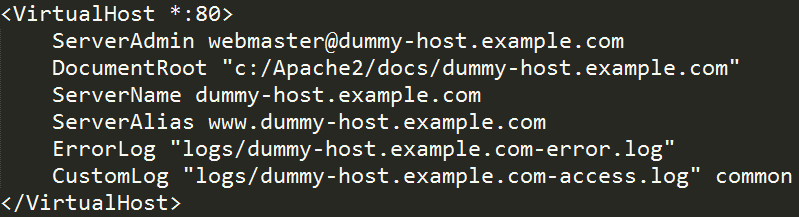
可以在一台Web服务器上指定多个虚拟服务器，可分别为每个虚拟服务器指定根目录。

a)在httpd.conf 中找到



去掉前面的#号注释，开启虚拟主机配置

b)打开conf/extra/httpd-vhosts.conf



DocumentRoot "D:/www/example"

ServerName "example.com "

ServerAlias "www.example.com"

其它项无需指定

c)修改DNS（hosts）文件

打开C:\Windows\System32\drivers\etc\hosts



注：修改hosts文件权限

d)重启Apache

### 3.4 配置局域网访问

首先需要关闭防火墙。

因为Apache默认不允许外部访问，需要修改配置：

在bin\apache\Apache2.2.21\conf文件中，搜索 onlineoffline tag - don't remove，将在234行附近的Allow from 127.0.0.1替换为Allow from all，保存后重启。

提示：获取本机IP，命令行输入ipconfig，IPV4即是。

## 4 PHP基础

### 4.1 PHP文件与注释

PHP文件以.php结尾，如下所示：

<?php

//单行注释

/\*

多行注释

\*/

echo "hello world";

?>

### 4.2 PHP变量

变量以$开头，后面跟着变量名，变量名可以包含字母/数字/下划线，不能以数字开头，区分大小写。

变量作用域：global local \*static

数据类型：字符串、整数、浮点数、布尔、数组、对象、NULL。

定义字符串时候，要注意引号的使用：

单引号：’’内部的内容只是作为字符串

双引号：””如果内部是PHP的变量，那么会将该变量的值解析

PHP的字符串连接符是：.

### 4.3 函数

##### 4.3.1 PHP函数使用

<?php

// 基础语法

function functionName() {

这里写代码

}

// 无参数 无返回值的函数

function sayhi(){

echo "Hello World";

}

// 有参数 无返回值的函数

function sayName($name){

echo $name.'你好哦';

}

// 调用

sayName('小狐狸');

// 有参数,参数有默认值的函数

function sayFood($food='西兰花'){

echo $food.'好好吃';

}

// 调用

sayFood('西葫芦');// 如果传入参数,就使用传入的参数

sayFood();// 如果不传入参数,直接使用默认值

// 有参数,有返回值的函数

function sum($a,$b){

return $a+$b

}

sum(1,2);// 返回值为1+2 = 3

?>

##### 4.3.2 常见函数

gettype() 检测数据类型

is\_string() 是否是字符

is\_array() 是否是数组

// foreach 循环

$colors = array("red","green","blue","yellow");

// 参数1为循环的对象,参数2会将对象的值挨个取出,直到最后

// 如果循环的是对象的话,输出的是对象属性的值

foreach ($colors as $value) {

echo "$value <br>";

}

内容输出函数：

echo：输出简单数据类型，如字符串、数值

print\_r()：输出复杂数据类型，如数组

var\_dump()：输出详细信息，如对象、数组

$arr =array(1,2,'123');

echo'123';

// 结果为 123

print\_r($arr);

// 结果为 Array ( [0] => 1 [1] => 2 [2] => 123 )

var\_dump($arr);

/\* 结果为

array

0 => int 1

1 => int 2

2 => string '123' (length=3)

\*/

### 4.4 文件引入

include 引入失败后程序继续执行

require 引入失败后程序终止执行

### 4.5 对象

// 定义最基础的类

class Fox{

public $name = 'itcast';

public $age = 10;

}

$fox = new $fox;

$name = $fox->name; // 对象属性取值

$fox->name = '小狐狸'; // 对象属性赋值

// 带构造函数的对象

class fox{

var $name = '小狐狸'; // 私有属性,外部无法访问

function Name(){ // 定义方法 用来获取属性

return $this->name;

}

function fox($name){ // 构造函数,可以传入参数

$this->name = $name

}

}

// 定义了构造函数 需要使用构造函数初始化对象

$fox = new fox('小狐狸');

// 调用对象方法,获取对象名

$foxName = $fox->Name();

### 4.6 PHP中的header

header()函数用来向客户端（浏览器）发送报头，如果中文无法显示，可以通过header设置编码：

header("content-type:text/html; charset=utf-8");

header设置浏览器跳转：

header('location:http://www.baidu.com');

header设置页面刷新：

header('refresh:3; url=http://www.xiaomi.com');

### 4.7 表单处理

##### 4.7.1 GET请求获取数据

在PHP中，通过$\_GET这个对象获取GET请求的参数。

表单请求代码：

<form action="01.php" method="get" >

<label for="">姓名:

<input type="text" name= "userName"></label>

<br/>

<label for="">邮箱:

<input type="text" name= "userEmail"></label>

<br/>

<input type="submit" name="">

</form>

表单name属性的是用来提供给服务端接收所传递数据而设置的

表单action属性设置接收数据的处理程序

表单method属性设置发送数据的方式

$\_GET获取代码：

<?php

echo "<h1>GET\_PAGE</h1>";

echo 'userName:'.$\_GET['userName'];

echo '<br/>';

echo 'userEmail:'.$\_GET['userEmail'];

?>

##### 4.7.2 POST请求数据获取

HTML表单只需要将请求方式改为POST即可。

<?php

echo "<h1>POST\_PAGE</h1>";

echo 'userName:'.$\_POST['userName'];

echo '<br/>';

echo 'userEmail:'.$\_POST['userEmail'];

?>

##### 4.7.3 请求错误处理

当我们直接访问POST或者GET的页面时，由于没有传递任何数据，会因为$\_GET、$\_POST不存在对应的key而报错，处理方式如下：

使用 array\_key\_exists(key,数组)函数来判断，参数1是要检测的key字符串，参数2是检验的数组。

if(array\_key\_exists('name', $\_GET)){

//如果有数据 再去读取

}else{

// 反之 可以执行一些 其他的逻辑

}

##### 4.7.4 文件上传

文件上传的html代码需要进行如下设置：

1 在html表单中需要 enctype=”multipart/form-data”

2 只能使用post方式，PHP使用$\_FILES获取

$\_FILES用法跟$\_GET,$\_POST类似,都是关系型数组

#\_FILE['key']:可以获取对应上传的文件,这里的key跟提交时的name相对应

#\_FILE['key']['name'] 可以获取上传的文件名

#\_FILE['key']['tmp\_name']可以获取上传的文件保存的临时目录

HTML代码示例：

<form action="03.fileUpdate.php" method="post" enctype="multipart/form-data">

<label for="">照片:

<input type="file" name = "picture" multiple="">

</label>

<br/>

<input type="submit" name="">

</form>

PHP代码：

sleep(5);// 让服务器休息一会

print\_r($\_FILES);

提交后会出现的现象：

1-点击提交后,服务器没有立即出现反应,而是休息了一会sleep(5)

2-在wamp/tmp目录下面出现了一个.tmp文件，且一会就被自动删除了

3-服务器返回的内容中,有文件的名字[name] => computer.png,以及上传文件保存的位置D:\wamp\tmp\php3D70.tmp

##### 4.7.5 文件保存

$\_FILES对象会先将文件保存在临时目录下，下面将演示如何将临时目录下的文件保存起来。

move\_uploaded\_file()这个函数可以移动文件。

Array ( [picture] => Array (

[name] => computer.png

[type] => image/png

[tmp\_name] => D:\wamp\tmp\php8913.tmp

[error] => 0 [size] => 5212 )

)

上传HTML代码依然如上。

PHP代码：

<?php

sleep(5);// 让服务器休息一会,方便我们查看上传的临时文件

// 第一个参数是 规定要移动的文件

// 第二个参数是 规定文件的新位置

move\_uploaded\_file($\_FILES['picture']['tmp\_name'], './upload/'.$\_FILES['picture']['name']);

?>

##### 4.7.6 设置上传文件大小

左键点击wamp任务栏图标，选择php，选择php.ini，对该文件进行修改。修改后需要重启wamp。

设置文件最大上传限制(值的大小可以根据需求修改)：

file\_uploads = On; 是否允许上传文件 On/Off 默认是On

upload\_max\_filesize = 32M; 上传文件的最大限制

post\_max\_size = 32M; 通过Post提交的最多数据

考虑网络传输快慢,这里修改一些参数：

max\_execution\_time = 30000; 脚本最长的执行时间 单位为秒

max\_input\_time = 600; 接收提交的数据的时间限制 单位为秒

memory\_limit = 1024M; 最大的内存消耗