# 运算符考点

**foo()和 @foo之间的区别**

考点是 php的运算符的错误控制符@

延申符的优先级

比较运算符

递增/递减运算符

逻辑运算符

## PHP的错误控制符

PHP支持一个错误运算符：@。当将其放置在一个PHP表达式之前，该表达式可能产生的任何错误信息都被忽略掉

## 延申考点：PHP运算符优先级

运算符优先级

递增/递减>!>算术运算符>大小比较>(不)相等比较>引用>位运算符(^)>位运算符(|)>逻辑与>逻辑或>三目>赋值>and>xor>or

括号的使用可以增加代码可读性，推荐使用

==和===的区别 ==是比较值的区别===不仅比较值还要比较类型的区别

等值判断（FALSE的七种情况都是等值的）

递增/递减运算符不影响布尔值

递减对NULL值没有效果

递增NULL的值为1

递增和递减在前就先运算符后返回，反之就先返回，后运算

**段路作用**

||和&&与or和and的优先级不同

$a = true || $b ==3;

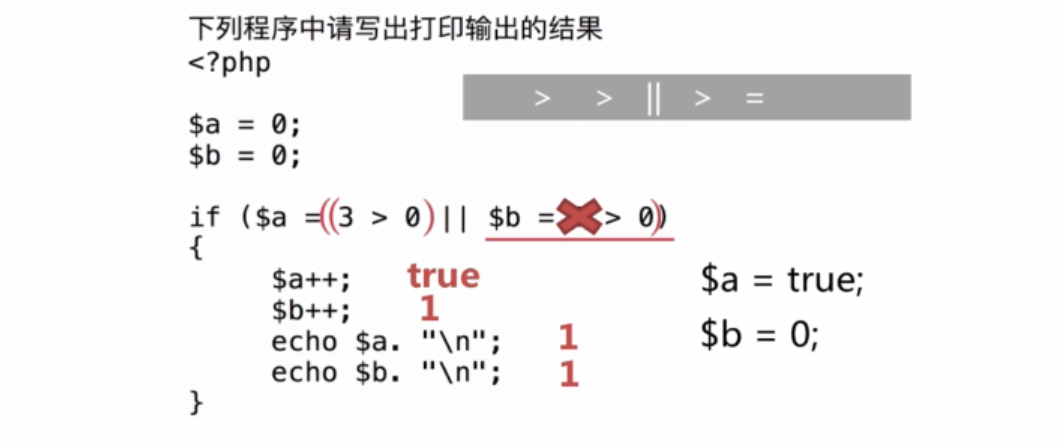
$b = false && $a==1;

|| && and or 它们的优先级分别是 && > || > and > or

$a = false || true;==>true ;||优先级高于=值所以先运行false || true 在整体返回结果

$b = false or true;==>false；or的优先级是最低的比=还低所以会先执行$b = false然后$b是false整体是true 最后是false

重点记忆递增/递减运算符的运算规则，逻辑运算符的短路效果，在看到逻辑运算符要多考虑优先级问题



这里=号的优先级是最低的其次是||，>号优先级最高所以运算的顺序是3>0是true所以运算到||符号时就不比较后面的$b了直接true赋给了$a

由于$a被运算到了所以是true而$b没有运算到所以$b的值本身的值0，而0++就等于1

# 流程控制考点

请列出3种PHP数组循环操作的语法，并注明各种循环的区别。

使用for循环

使用foreach循环

使用while,list(),each() 组合循环

for循环只能遍历索引数组，foreach可以遍历索引和关联数组，联合使用list(),each()和while循环同样可以遍历索引和关联数组。

while,list(),each()组合不会reset()

foreach 遍历会对数组进行reset()操作

分枝结构：

If ..... elseif

在elseif语句中只能有一个表达式为true,即在elseif语句中只能有一个语句块被执行。多个elseif从句是排斥关系。

使用elseif语句有一个基本原则，总把优先范围小的条件放在前面处理。

例如：像下面的代码如果其中有一个满足就会执行其中一个代码，后面的就不会在作判断了。如果都不满足就执行else。

If(exp1){

}elseif(exp2){

}elseif(exp3){

}else{

}

**switch .....case...**

和if不同的是,switch后面的控制表达式的数据类型只能是整形，浮点型或者字符串。

//如果在switch外有一层for想让continue作用于for跳出本次循环就使用continue2表示跳出两层，第一层是switch第二层for。

for(){

switch($var){

case ..;

continue; 如果在switch里这样加continue它的作用等价于break

break;

case ..;

break;

case..;

break;

default...;

break;

}

}

switch ...case 会生成跳转表，直接跳转到对应case

效率：如果条件比一个简单的比较复杂得多或者在一个很多次的循环中，那么用switch语句可能会块一些。

理解循环内部机制，更易于记忆foreach的reset特性，分枝结构中理解了switch ....case的执行步骤也就不难理解为什么效率高了。

**PHP中如果优化多个if...elseif语句的情况？**

1、可以把可能性较大的尽量往前挪。

2、如果判断的是一个比较复杂的内容而且判断的是整形，浮点型或者是字符串类型就可以使用switch，这样会有一个整体效率上的提升。

# 变量的作用域

变量的作用域也称变量的范围，变量的范围即它定义的上下文背景（也是它的生效范围）。大部分的PHP变量只有一个单独范围。这给单独的范围跨度同样包含了include和require引入的文件。

global 关键字

$GLOBALS及其他超全局数组。

静态变量仅在局部函数域中存在，但当程序执行离开此作用域时，其值并不会消失

**static关键字**

1. 仅初始化一次
2. 初始化时需要赋值
3. 每次执行函数该值会保留
4. static修饰的变量是局部的，仅在函数内部有效
5. 可以记录函数的调用次数，从而可以在某些条件下终止递归

例 ：

Function mgFunc(){

static $a=1;//$a不会被释放调

echo $a++;

}

myFunc();-->1 #每执行一次$a就累加一次

myFunc();-->2

myFunc();-->3

这其实我们可以用来判断递归的执行次数，如当递归达到多少次以后就停掉函数

$count =5;

function get\_count(){

static $count; #在调用第二次的时候不会在执行初始化$count

return $count++;

#由于优先级的关系第一次执行到这里的时候是先赋值后++所以第一次调用时$count没有值所以null，到调用第二次输出的1其实是第一次调用时++的结果，但是此时$count在内存中已经是2。

而如果是 return ++$count的话结果就会完全不一样，先++后赋值给return结果就是1和2而不是null和1。

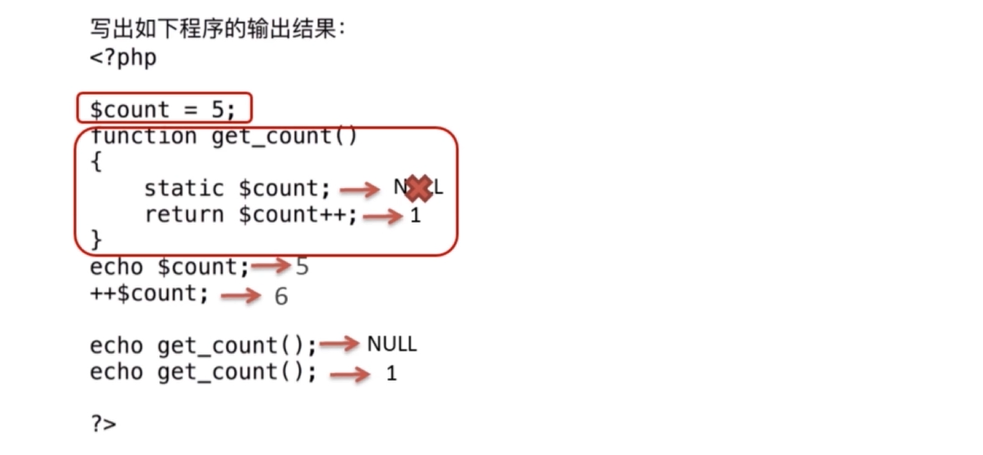
}

echo $count;

++$count;

echo get\_count();

echo get\_count();



默认情况下，函数参数通过值传递

如果希望允许函数修改它的值，必须通过引用传递参数

$a=1;

function myFun($a){

$a=2;

}

myFunc($a);

echo $a;

在这个案例中$a在函数体中被改变了值，这时外部的$a不会受影响。

但如果是下面的情况时：

$a=1;

function myFun(&$a){ //$a使用了&引用传递

$a=2;

}

myFunc($a);

echo $a;

只有当$a传进函数体内部的时候使用&引用传递外面的$a才会受影响。

值通过使用可选的返回语句 ( return ) 返回

可以返回包括数组和对象的任意类型

返回语句会终止函数执行，将控制权交回函数调用处。

省略return,返回值为NULL,不可有多个返回值

从函数返回一个引用，必须在函数声明和指派返回值给一个变量时都使用引用运算符 &

function &mgFunc(){

static $b=10;

return $b;

}

$a=myFunc(); -->10此时是正常调用所以返回值是10

$a=&myFunc(); -->加&符号后此时 $a和 $b之间就变成了互为引用；即$a指向$b的内存空间

$a=100; -->这时给$a赋值100也相当于给$b 赋值100；

echo myFunc(); -->所以最后的结果是输出100

## 外部文件的导入

Include/require 语句包含并允许指定文件

如果给出路径名安装路径来找，否则从include\_path 中查找如果include\_path中也没有，则从调用脚本文件所在的目录和当前工作目录下寻找

当一个文件被包含时，其中所包含的代码继承了include所在行的变量范围

加载过程中找到文件则include结构会发出一条警告；这一点和require不同，后者会发出一个致命错误

require在出错时产生E\_COMPILE\_ERROR级别的错误。换句话说将导致脚本终止而include只产生警告(E\_WARNING),脚本会继续运行。

require(include)/require\_one(include\_once)唯一区别是PHP会检查该文件是否已经被包含过，如果是则不会在次包含。

**时间日期函数**

**date(),strtotime(),mktime(),time(),microtime(),date\_default\_timezone\_set()**

**IP处理函数**

**Ip2long(),long2ip();**

**打印处理**

**print(),printf,print\_r(),echo,sprintf(),var\_dump(),var\_export()**

**序列化及反序列化函数**

**serialize() , unserialize()**

**字符串处理函数**

**Implode(),explode(),join(),strrev(),trim(),ltrim(),rtrim(),strstr(),number\_format()....**

**数组处理函数**

**array\_keys(),array\_values(),array\_diff(),array\_intersect(),array\_merge(),array\_shift,array\_unshift(),array\_pop(),array\_push(),sort(),array\_rand等**

**着重记忆PHP函数的定义相关内容，理解变量作用域，静态变量，函数的参数和返回值的相关内容，重点记忆我们总结的内置函数。**

$var1 = 5;

$var2= 10;

function foo(&$my\_var){

global $var1; -->5 取全局变量 var1

$var1 +=2; -->7 $var1加2等于7

$var2 = 4; -->4 $var2没有定义global取全局变量所以，在这里定义只能算局部变量所以是4

$my\_var += 3; -->8 $my\_var使用了引用传递$my\_var在外部定义的是5引用传递进来后加3等于8。同时也改变了外面的$my\_var值

return $var2;

}

$my\_var = 5;

echo foo($my\_var).'\n';

echo $my\_var.'\n';

echo $var1; -->7 在函数体内使用global调用并修改过也会影响外面的值所以是7

echo $var2; -->10 $var2在函数体中只能算是局部变量所以在函数内作的任何修改不会形象到外面

$bar = 'foo';

$my\_var = 10;

echo $bar($my\_var).'\n'; -->4 调用foo函数

# 正则表达式考察点

至少写出一种验证139开头的11位手机号码的正则表达式。

$str = '13988888888';

$pattern = '/^139\d{8}$/';

preg\_match($pattern,$str,$match);

var\_dump($match);

正则表达式的作用：分割，查找，匹配，替换字符串

分割符：正斜线(/),hash符号(#)以及取反符号( ~ )

通用原子：\d,\D,\w,\W,\s,\S

元字符：.\* ? ^ $ + {n} {n,} {n,m} [] () [^] | [-]

模式修正符：i m e s U x A D u

后向引用

$str=’<b>abc</b>’;

$pattern=’<b>(.\*)<\/b>/’;

preg\_replace($pattern,’\\1’,$str);//把前面匹配到的放到这，用\1表示所以角后向引用

贪婪模式

”.\*?”表示取消贪婪模式匹配得到的是每一个b标签的内容。如果只是.\*表示匹配以第一个<b>标签开头然后一值到脚本结束的</b>而中间无论出现多少个</b>都把他当成内容匹配。

$str='<b>abc</b><b>bcd</b>';

$pattern='/<b>.\*?<\/b>/';

preg\_replace\_all($pattern,'\\1',$str);

$pattern='/<b>.\*</b>/U'//第二种取消贪婪的方法

正则表达式PCRE函数：preg\_match(),preg\_match\_all(),preg\_replace(),preg\_split()

中文匹配：UTF-8汉字编码范围0x4e00-0x9fa5,在ANSI(gb2312)环境下，0xb0-0xf7,

0xa1-0xfe

例：

//匹配中文字符串方法

$str='中文';

$pattern = '/[\x{4e00}-\x{9fa5}]+/u';//匹配UTF-8编码的中文字符

$pattern = '/['.chr(0xb0).'-'.chr(0xf7).']['.chr(0xa1).'-'.chr(0xfe).']/';//匹配gbk编码的中文字符

preg\_match($pattern,$str,$match);

var\_dump($match);

UTF-8要使用u模式修正符使模式字符串被当成UTF-8,在ANSI(gb2312)环境下，要使用chr将Ascii码转换为字符

请写一个正则表达式，取出页面中所有img标签中的src值

$str = '<img alt="高清无码" id="av" src="av.jpg" />';

$pattern = '/<img.\*?src=".\*?".\*?\/?>/i'; //.匹配任意字符 \*零次或多次?零次或一次

preg\_match($pattern,$str,$match);

var\_dump($match);

结果：

**array** *(size=1)*

0 => string '<img alt="高清无码" id="av" src="av.jpg" />' *(length=47)*

**取出src的值在src后面的匹配字符加括号**

$str = '<img alt="高清无码" id="av" src="av.jpg" />';

$pattern = '/<img.\*?src="(.\*?)".\*?\/?>/i'; //.匹配任意字符 \*零次或多次?零次或一次

preg\_match($pattern,$str,$match);

var\_dump($match);

**结果：**

**array** *(size=2)*

0 => string '<img alt="高清无码" id="av" src="av.jpg" />' *(length=47)*

1 => string 'av.jpg' *(length=6)*

# **文件及目录处理**

不断在文件hello.txt头部写入一行”Hello World“字符串，要求代码完整

//打开文件

//将文件的内容读取出来，在开头加入Hellow wrold

//将拼接好的字符串写回到文件当中

$file = './hellow.txt';

$handle = fopen($file,'r');

$content = fread($handle,filesize($file));

$content = 'Hellow World'.$content;

fclose($handle);

$handle = fopen($file,'w');

fwrite($handle,$content);

fclose($handle);

通过PHP函数的方式对目录进行遍历，写出程序

$dir = './test';

//打开目录

//读取目录当中的文件

//如果文件类型是目录，继续打开目录

//读取子目录的文件

//如果文件类型是文件，输出文件名称

//关闭目录

//

function loopDir($dir){

$handle = opendir($dir);

while(false!==($file = readdir($handle))){

if($file != '.' && $file !='..'){

echo $file."\n";

if (filetype($dir. '/'.$file) == 'dir'){

loopDir($dir. '/'.$file);

}

}

}

}

loopDir($dir);

fopen()函数

用来打开一个文件，打开时需要指定打开模式

**打开模式：**

r：只读方式打开，并且将文件的指针指向文件的开头

r+：读写方式打开，并且将文件的指针指向文件的开头

w：写入方式打开，并且将文件的指针指向文件的开头，并且将文件清空

w+：读写模式，如果文件部存在自动创建

a：追加的写入方式，将文件的指针指向文件末尾，如果文件不存在则创建

a+：读写方式，读写追加，文件指针指向末尾，如果文件不存在则创建

x：在创建的时候以写入的方式打开，并且将文件指针指向文件开头，如果文件以及存在会报一个warning从错误并且fopen返回一个false如果文件不存在才会帮你去创建

x+：创建并以读写的方式打开

b：打开一个二进制文件

t：Windows 下提供了一个文本转换标记（"t"）可以透明地将 \n 转换为 \r\n

b和t是配合前面的4大类来使用的

写入函数

fwrite() //最常用

fputs()

读取函数

fread()

fgets() //获取一行

fgetc() //获取一个字符

关闭文件函数

fclose()

不需要fopen()打开的函数

file\_get\_contents()

file\_put\_contents()

其他读取函数

file() //把整个文件读取到一个数组里面

readfile() //把内容读取出来并且把内容读取到缓冲区

访问远程文件

开启allow\_url\_fopen,HTTP协议连接只能使用只读，FTP协议可以使用只读或者只写

## 目录操作函数

名称相关：basename(),dirname(),pathinfo()

目录读取：opendir(),readdir,closedir(),rewinddir()

目录删除：rmdir() 这个函数只能删除空目录。

目录创建：mkdir()

文件大小：filesize()

目录大小只能遍历目录然后把文件大小加起来

目录大小：

disk\_free\_space() 磁盘大小剩余空间

disk\_total\_space() 总共大小

文件拷贝:copy()

删除文件：unlink()

文件类型：filetype()

重命名文件或者目录：rename()可以重命名还可以移动目录

文件截取：ftruncate()

文件属性：

file\_exists() 判断文件是否存在

is\_readable()是否可读

is\_writable()是否可写

is\_executable()是否可执行

filectime() inode 修改时间

fileatime() 访问的时间

filemtime() 整个修改时间

**文件锁：**

flock()

文件指针：ftell(),fseek(),rewind()

# 会话控制技术

简述cookie和session的区别及各自的工作机制，存储位置等，简述cookie的优缺点。

为什么要使用会话控制技术？

通过GET参数传递

session 同步到分布式服务器

session\_set\_save\_handler(); 设置用户自定义会话存储函数

存储到MySQL,Memcache,Redis等

# 面向对象

请写出PHP类权限控制修饰符

**魔术方法**

\_\_construct(),\_\_destruct(),\_\_call(),\_\_callStatic(),\_\_get(),\_\_set(),\_\_isset(),\_\_unset(),\_\_sleep(),\_\_wakeup(),\_\_toString(),\_\_clone()

设计模式

常见设计模式：工厂模式，单列模式，注册树模式，适配器模式，观察者模式和策略模式

在开发当中用户那些设计模式，设计模式的实现原理是什么

请写出PHP的构造函数和析构函数

\_\_construct()构造函数

\_\_destruct()析构函数

# 网络协议

HTTP/1.1中，状态码 200 301 304 403 404 500 的含义

OSI七层模型

HTTP协议的工作特点和工作原理

HTTP协议常见请求/响应头和请求方法

HTTPS协议的工作原理

常见网络协议含义及端口

HTTP协议状态码

五类响应：1XX,2XX,3XX,4XX,5XX

1 开头的叫做信息类状态码，表示接收的请求正在处理

2 开头的表示成功状态码，表示请求正常处理完毕了

3 开头的叫做重定向，表示需要附加操作以完成请求

4 开头的表示客户端错误，表示服务器无法处理请求

5 开头的代表服务器端错误状态码，表示服务器处理请求时出错了

常见状态码：

200 表示从客户端到服务器的请求被正常处理了。成功

204 服务器接收的请求已成功处理，但返回的响应报文中不含实体的主体部分

206 表示客户端进行了部分的内容请求，服务端成功的执行了这部分的get请求

301 代表跳转，永久性重定向。请求的资源已经永久性的分配了新的URL，以后应使用现在所指的url

302 临时性重定向，请求的资源已经分配是新的URL希望用户能使用本次的url来访问

303 由于请求对应的资源存在着另一个url应使用get方法定向获取请求的资源

304 客户端发送附带的请求时服务器端允许请求访问资源，单一发生请求为满足条件的情况后直接返回304

307 临时重定向，和302相同

400 请求报文中存在语法错误，当错误发生时，需修改内容后在次发送请求

401 发送的请求需要有http认证的认证信息

403 被请求的资源访问被服务器拒绝了

404 服务器找不到请求的资源

500 服务器端在执行请求时发生了错误

503 代表服务器超负载或是停机维护

**OSI七层模型**

第一层：物理层

建立，维护，断开物理连接

第二层：数据链路层

建立逻辑连接，进行硬件地址寻址，错误校验等功能

第三层：网络层

进行逻辑地址寻址，实现不同网络之间的路径选择。

第四层：传输层

定义传输数据的协议端口号，以及流程和差错校验

协议有：TCP UDP ,数据包一旦离开网卡即进入网络传输层

第五层：会话层

建立，管理，终止会话**Telnet**

第六层：表示层

数据的表示，安全，压缩。

第七层：应用层

网络服务与最终用户的一个接口

协议有：HTTP FTP TFTP SMTP SNMP DNS TELNET HTTPS POP3 DHCP

**HTTP协议的工作特点和工作原理**

工作特点

基于B/S模式

通信开销小、简单快速、传输成本低

使用灵活、可使用超文本传输协议

节省传输时间

无状态

**工作原理**

客户端发送请求给服务器，创建一个TCP连接、指定端口号，默认80、连接到服务器、服务器监听浏览器请求、一旦监听到客户端请求、分析请求类型后，服务器会像客户端返回状态信息和数据内容

**HTTP协议常见请求/响应头和请求方法**

Content-Type:请求与实体对应的MIME信息、

Accept：指定客户端能接受的内容类型

Origin：最初的请求来源于哪，主要用于post请求

Cookie：http请求发起时发送给服务端的COOKIE的值

Cache-Control：指定请求和响应的缓存机制

User-Agent：用户信息

Referrer：上级请求路径

X-Forwarded-For：请求端真实的IP

Access-Control-Allow-Origin：允许特定的域名来访问

Last-Modified：请求资源的最后响应时间

**HTTP协议常见请求/响应头和请求方法**

GET,POST,HEAD,OPTIONS,PUT,DELETE,TRACE

http协议的GET和POST请求方法的区别

1. 刷新是get数据没有变化，而post则会重新提交
2. Get可以收藏为书签，post不可能被收藏为书签
3. Get请求可以被浏览器缓存，post不能被浏览器缓存
4. get请求时编码类型是
5. 如果传递大量数据、比如大文件的上传下载，只能使用POST请求，GET请求的URL有长度限制，一般不超过1kb；
6. POST安全性较好，如果包含敏感信息比如用户登录的账号密码，使用POST请求；

**HTTPS协议的工作原理**

HTTPS是一种基于SSL/TLS的HTTP协议，所有的HTTP数据都是在SSL/TLS协议封装之上传输的。

HTTPS协议在HTTP协议的基础上，添加了SSL/TLS握手以及数据加密传输、也属于应用层协议。

**常见网络协议含义及端口**

FTP :文件传输协议，上传或下载文件默认端口 21

Telnet：用于远程登录的端口用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于dos模式下的通信服务默认端口23

SMTP：简单邮件传输协议默认端口是25

POP3：和SMTP对应主要用于接收邮件默认端口110

HTTP：超文本传输协议默认端口80

DNS：用于域名解析服务默认端口53

我们常见的HTTP协议、TCP协议分别位于OSI网络模型的第几层？

TCP在第四层：传输层、HTTP在地区层应用层

# 开发环境及配置

您是否使用过版本控制器软件？如果有您使用的版本控制软件的名称是什么？

**版本控制软件**

**PHP运行原理**

**PHP的常见配置项**

**版本控制软件**

**集中式和分布式**

集中式：中间有一个中央的服务器所有的客户机都把我们的版本都上传到中央服务器，也就是说版本中在中央服务器里保存。一旦中央服务器挂掉客户机就没有办法区下载和上传版本了。

**CVS和SVN**

**Git是分布式**

**PHP运行原理**

**Nginx+PHP-FPM**

**CGI:**早期它只能处理一些简单的html的静态文件。随着计算的发展我们出现了动态的语言比如php或python :在这当中如果说我们要区处理PHP 我们是不是要交给PHP解析器来处理，但是PHP处理完只会又怎么跟我们的webserver进行通信呢，所以在这里为了解决不同的语言处理器与web进行通信出现了CGI协议只要你按照CGI协议去编写程序就能够实现语言解析器与我们web的通信比如说我们的PHP cgi程序。在这当中它其实就是我们的php解析器跟我们的webserver当中的一个协议桥梁

FastCGI:CGI虽然是解决了php跟web通信的问题但是它的效率很低，因为webserver每收到一个请求都会fock一个cgi进程然后请求结束的时候在kill掉这个进程这时我们如果有十万个几十万个或是一百万个请求的话我们就fock一百万这样的进程然后用完以后在kill掉这样本身来说非常浪费我们的资源所以此时fastcgi就出现了。它主要是以cgi的改良版本出现的

fastcgi每次请求完以后不会kill掉进程而是保留这个进程使这个进程可以一次处理多个请求这样的话就不用每次都从新fock一个进程了。大大提升了我们的处理效率这就是fastCGI

PHP-FPM:FastCGI Process Manager：FastCGI进程管理器 PHP-FPM是fastcgi的进程管理器所以在这php-fpm本身是fastcgi的一个实现，并且提供了进程管理的功能进程包括master进程和worker进程两种进程；master进程只有一个负责监听端口接收来自webserver的请求

worker进程一般会有多个具体数量会在fpm的配置当中来定义每一个进程内部都会嵌入到

一个php解析器，php代码真正执行的地方。也就是说morker是处理我们的代码而master主要是负责监听我们的端口跟我们的webserver来进行接收请求。master默认监听端口是9000然后通过nginx的反响代理，代理9000端口所以在这来说我们可以完成我们fpm相关的处理

**PHP的常见配置项**

Register\_globals:注入变量，当我们打开时各种变量对会被注入代码。建议一定关闭不用用

allow\_url\_fopen:是否允许打开远程文件

Allow\_rul\_include:是否允许远程包含文件

Date.timezone:设置时区

Display\_errors:是否显示错误；在开发时是要开启的而在生产环境下一定要关闭

Error\_reporting:显示错误的级别设置

Safe\_mode:是否开启安全模式

Upload\_max\_filesize:上传的最大的文件大小是多少

Max\_file\_uploads:上传最大文件数量是多少

Post\_max\_size:提交post数据的最大大小是多少

请简述CGI、FastCGI和PHP-FPM的区别

cgi是为了联系webserver跟php解析器当中的一个协议

fastCGI 是cgi改良的版本

Php-fpm:进程管理器