**常见的经典面试题**

**什么是引用变量？在PHP当中，用什么符合定义引用变量？**

**考官考点：**

**PHP的引用变量的概念及定义方法**

在PHP中引用意味着用不同的名字访问同一个变量内容。

定义方式 ：使用 & 符号

**延申：PHP引用变量的原理**

**//定义一个变量**

$a = range(0,1000);//从0到1000的数组变量

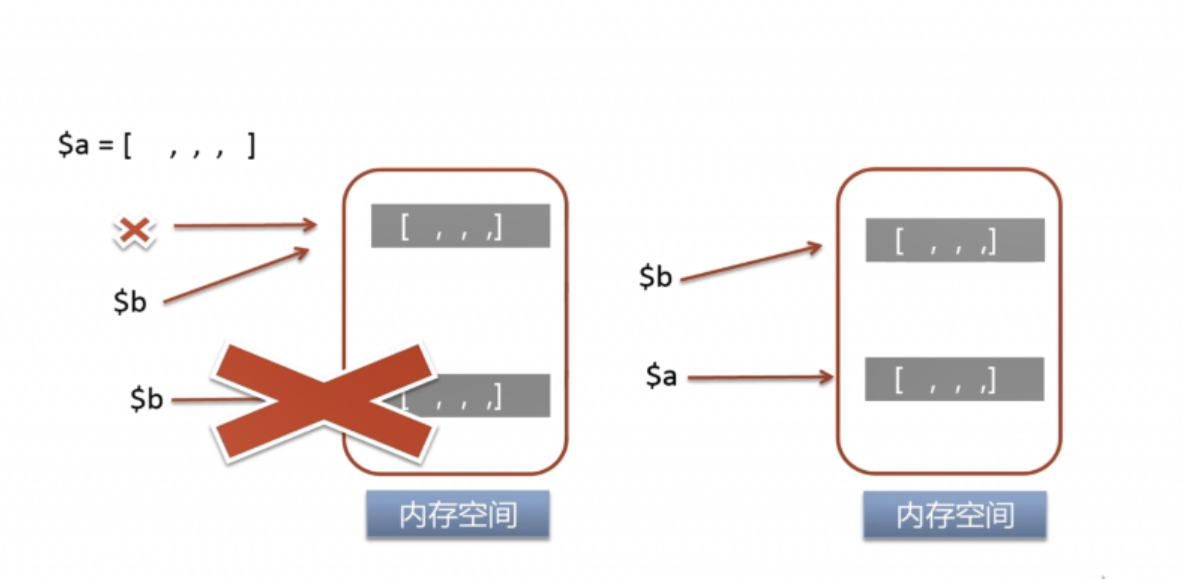
//定义变量b ，将a变量的值赋值给b

//COW Copy On Write

$b = $a

//对a 进行修改

$a = range(0 ,1000)



从图中我们可以看到当第一个$a变量定义时系统会给$a开辟一个内存空间，但是当$a赋值给$b时系统不是给$b开辟一个新的空间。而是让$b指向$a的内存空间，相当于$b调用了$a的内存空间。只有当$a重新被赋值的时候系统就会呈现右边的样子，系统会给$a和$b分别创建一个内存空间。

**//定义一个变量这次当$a给$b赋值时在$a前加个&符情况就不一样了**

$a = range(0,1000);//从0到1000的数组变量

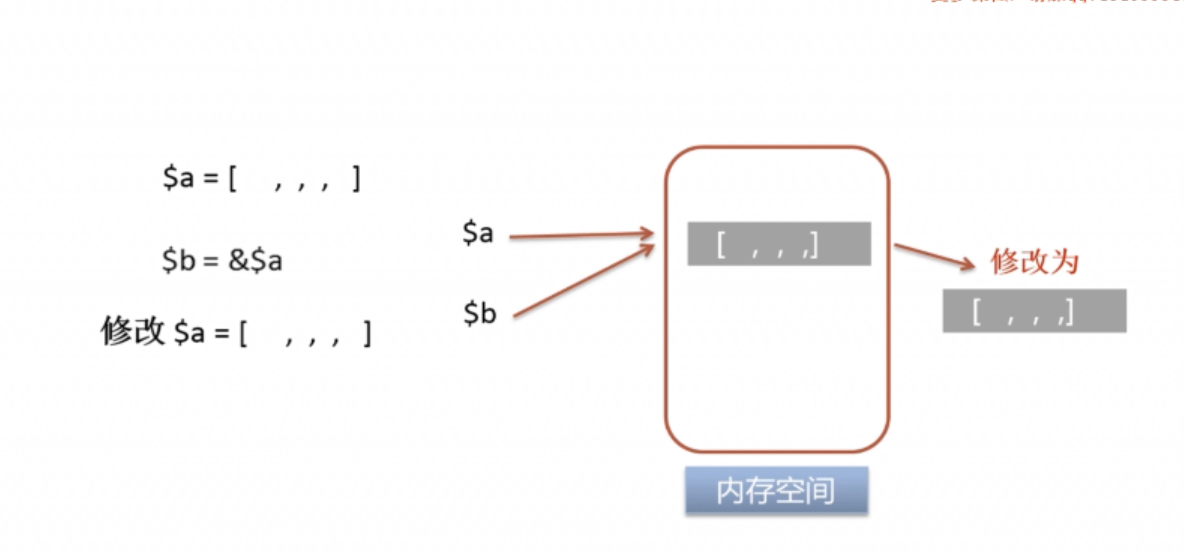
//定义变量b ，将a变量的值赋值给b

//COW Copy On Write

$b = &$a

//对a 进行修改

$a = range(0 ,1000)



如果所示，当我们给$a加上引用符号时情况就变成了$a被修改之后$a变量也会指向最初系统给$a开辟的内存空间。系统不会给$a从新开辟一个新的空间，就算是被修改也是修改的以前的空间的值。这就算引用的好处，$a和$b会永远指向同一个空间。

**我们用 xdebug\_debug\_zval 函数查看以下$a在内存中的变化**

$a=range(0,3);

xdebug\_debug\_zval('a');

a:

*(refcount=1, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

*refcount=1 //表示内存空间，1表示有一个变量指向这个空间*

*is\_ref=0 //表示，是不是引用，0表示不是引用*

**当我们把变量$a赋值给$b时情况是：**

$a=range(0,3);

xdebug\_debug\_zval('a');

$b=$a;

xdebug\_debug\_zval('a');

a:

*(refcount=1, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

a:

*(refcount=2, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

这时我们还是打印a我们可以看到*refcount已经变成了二，表示有两个变量同时指向这个内存空间。*

**而当我们在定义一个$a时看看是什么情况**

$a=range(0,3);

xdebug\_debug\_zval('a');

$b=$a;

xdebug\_debug\_zval('a');

$a=range(0,3);

xdebug\_debug\_zval('a');

a:

*(refcount=1, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

a:

*(refcount=2, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

a:

*(refcount=1, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

我们看到新定义的$a的*refcount* 的值又变回了1因为系统重新给$a开辟了新的内存空间，而不是引用上一个$a的内存空间。

**当$a使用 & 引用之后**

$a = range(0,3);

xdebug\_debug\_zval('a');

$b = &$a;

xdebug\_debug\_zval('a');

$a = range(0,3);

xdebug\_debug\_zval('a');

a:

*(refcount=1, is\_ref=0)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

a:

*(refcount=2,* *is\_ref=1)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

a:

*(refcount=2, is\_ref=1)*,**array** *(size=4)*

0 => *(refcount=1, is\_ref=0)*,int 0

1 => *(refcount=1, is\_ref=0)*,int 1

2 => *(refcount=1, is\_ref=0)*,int 2

3 => *(refcount=1, is\_ref=0)*,int 3

可以看到从第二个打印a开始 refcount和is\_ref都分别变成了2和1，表示有$a变量指向的空间一共有两个变量。Is\_ref=1表示使用了系统引用。从这就可以看出$a和$b只用了一个空间，而$a一直使用的是最初的空间。

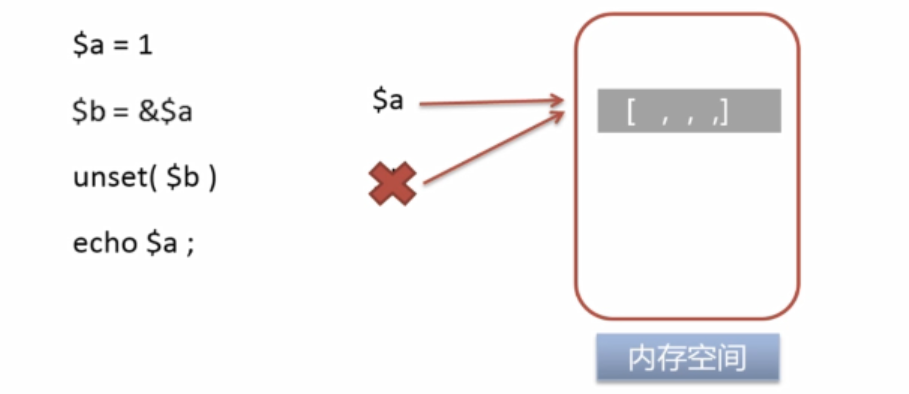
**Unset 只会取消引用，不会销毁空间。**

$a = 1;

$b = &$a;

unset($b);

echo $a."\n";



如果是unset是销毁空间的话那此时$a就是不存在值会显示未定义的状态，但是浏览器打印的结果是 1。上图是$a和$b在内存中的状态表现。

**对象在php中本身就是引用传值**

class Person{

public $name = "zhangsan";

}

$p1 = new Person;

xdebug\_debug\_zval('p1');

$p2 = $p1;

xdebug\_debug\_zval('p1');

$p2->name="lisi";

xdebug\_debug\_zval('p1');

p1:

*(refcount=1, is\_ref=0)*,**object**(*Person*)[*1*]

*public* 'name' => *(refcount=2, is\_ref=0)*,string 'zhangsan' *(length=8)*

p1:

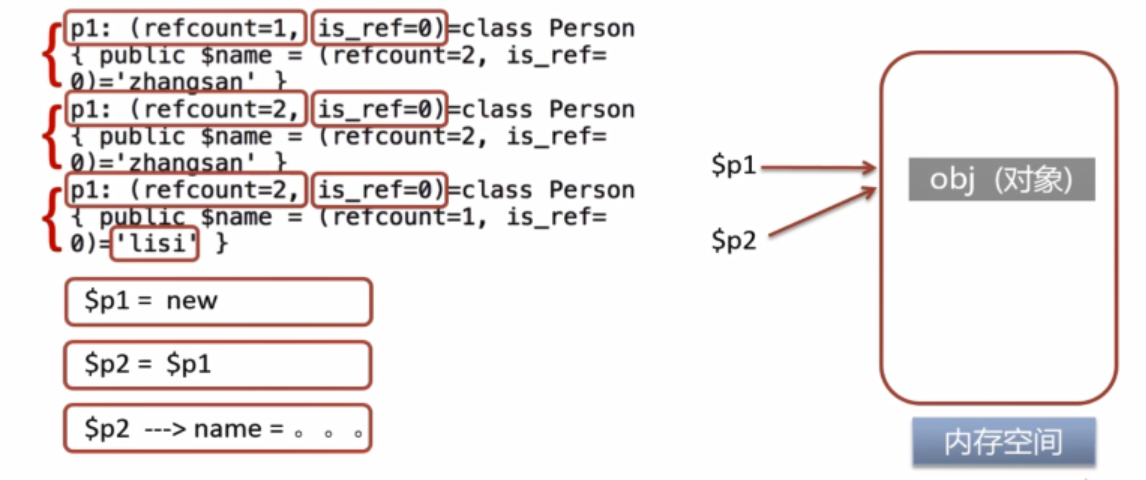
*(refcount=2, is\_ref=0)*,**object**(*Person*)[*1*]

*public* 'name' => *(refcount=2, is\_ref=0)*,string 'zhangsan' *(length=8)*

p1:

*(refcount=2, is\_ref=0)*,**object**(*Person*)[*1*]

*public* 'name' => *(refcount=1, is\_ref=0)*,string 'lisi' *(length=4)*



首先第一步的和第二步的时候可以看到都没有什么问题！第一步还没有p1还没有赋值，所以refcount值是1指向p1空间的只有一个变量。第二步时p1赋值给了p2 而refcount值就变成了2表示有两个变量指向p1空间。但是到第三步时我们给p1变量重新赋值了而且name变量已经变成了lisi，表示赋值已经成功。本来第三步的p1如果重新开辟了空间的话refcount 的值应该是1不是2。这也说明了系统并没有给p1重新开辟一个内存空间而是修改了p1原先的内存空间。

**写下如下程序的输出结果**

$data=['a','b','c'];

foreach($data as $key => $val){

$val = &$data[$key];

var\_dump($data);

}

var\_dump($data);

结果是：

F:\phpStoudy\PHPTutorial\WWW\index.php:8:**array** *(size=3)*

0 => string 'a' *(length=1)*

1 => string 'b' *(length=1)*

2 => string 'c' *(length=1)*

F:\phpStoudy\PHPTutorial\WWW\index.php:8:**array** *(size=3)*

0 => string 'b' *(length=1)*

1 => string 'b' *(length=1)*

2 => string 'c' *(length=1)*

F:\phpStoudy\PHPTutorial\WWW\index.php:8:**array** *(size=3)*

0 => string 'b' *(length=1)*

1 => string 'c' *(length=1)*

2 => string 'c' *(length=1)*

F:\phpStoudy\PHPTutorial\WWW\index.php:10:**array** *(size=3)*

0 => string 'b' *(length=1)*

1 => string 'c' *(length=1)*

2 => string 'c' *(length=1)*

因为第一次循环的时候&$data[$key]; 赋值还是原先$val的值所以第一次循环没有改变任何值。因为$data[0]的值是a而$val的值也是a

第二次循环时$val=a被$data[1]改变了内存指向，$val指向了b所以$data的值变成了b,b,c

第三次循环时$val=b被$data[2]改变了内存指向，$val指向了c所以$data的值变成了b,c,c

第三次循环$val=b指的是第二个位置的b所以只有第二个位置的b被改变，而第一个不受影响。

**要求写出jquery中，可用处理AJAX的几种方法。**

**写出进可能多的Linux命令**

**写出三种以上MySQL数据库存储引擎的名称**

**编写一个在线留言本，实现用户的在线留言功能**

**谈谈你对MVC的认识介绍几种目前比较流行的MVC框架**

**请写出常见的排序算法**

**PHP如何解决网站大流量与高并发的问题**