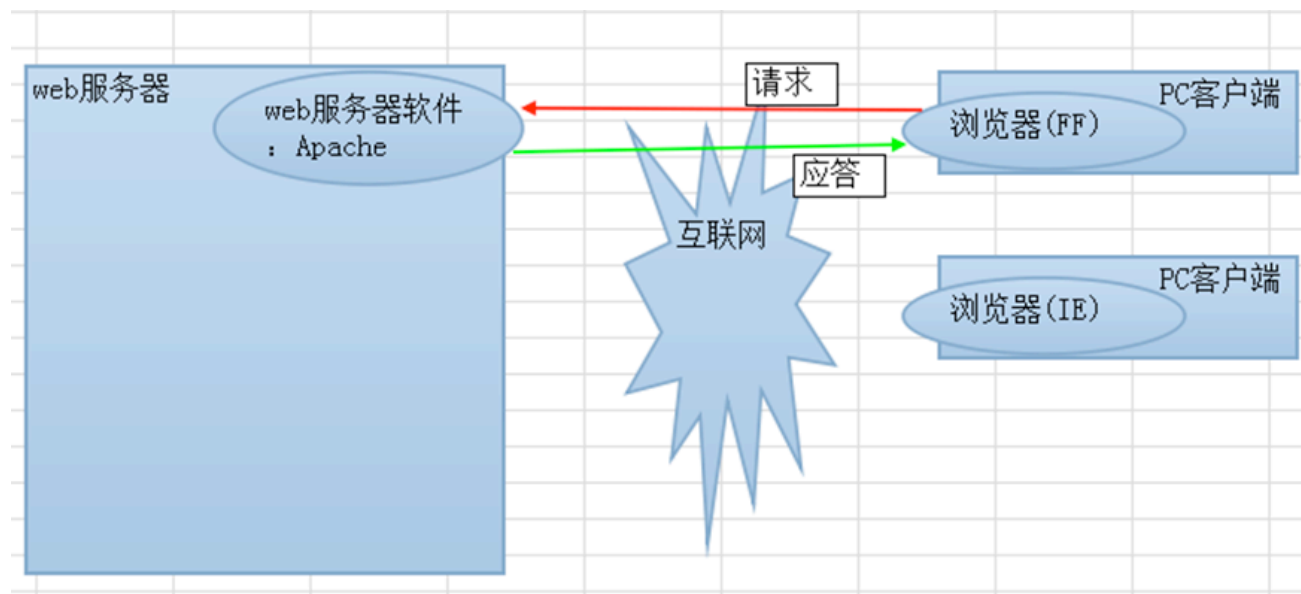


昨日回顾.....	2
php 基本语法形式.....	4
php 的标记符.....	4
php 的区分大小写特性.....	5
一条语句使用一个分号结束.....	5
php 的注释.....	6
变量.....	7
基本理解.....	7
变量的基本操作.....	7
赋值：	7
取值：	7
判断 isset(变量名):	8
删除 unset(变量名):	8
变量命名规则.....	8
基本规则——保证程序的正确性.....	8
行业规则——保证程序的可读性.....	9
变量的传值方式.....	9
值传递.....	9
引用传递.....	9
可变变量：	10
预定义变量.....	11
综述.....	11
\$_POST 变量.....	12
\$_GET 变量.....	13
\$_REQUEST 变量.....	15
\$_SERVER 变量	17
\$GLOBALS 变量	18

昨日回顾

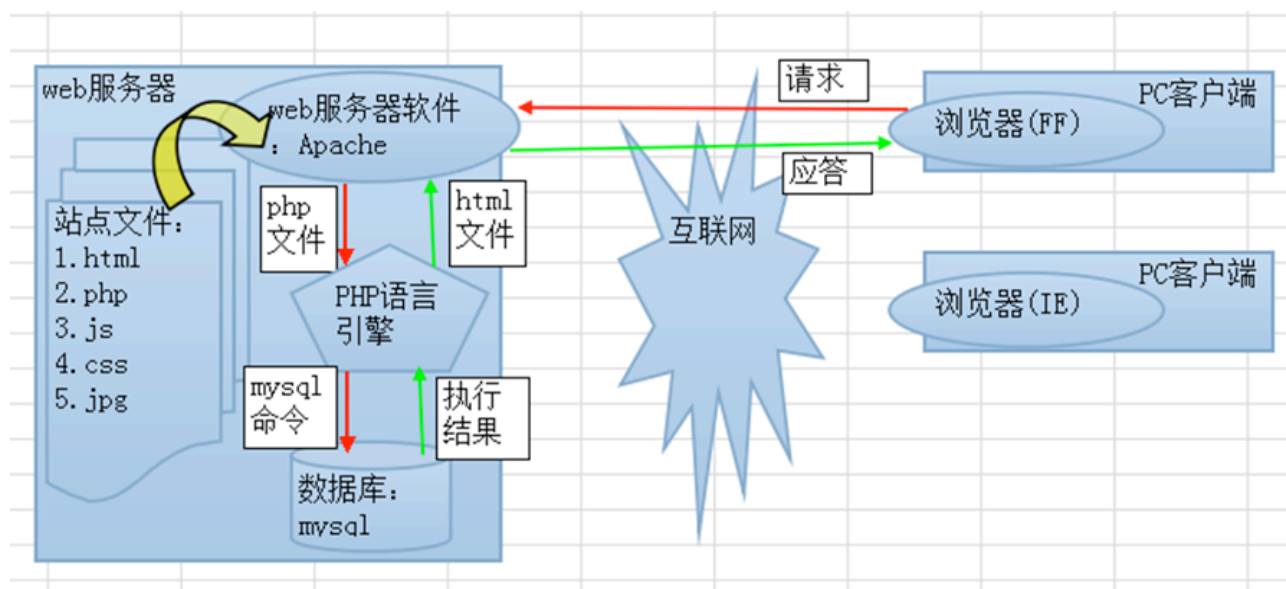
php 运行环境

web 软件的运行模式



web 运行流程

php 网页运行原理、



配置 apache 以运行 php 程序:

目标 1: `echo "abc";`

在 apache 的主配置文件 `httpd.conf` 中, 添加 2 行:

```
LoadModule php5_module "php 语言包所在目录/php5apache2_2.dll"
```

```
AddType application/x-httpd-php .php .php3 .phtml
```

补充一个检测 apache 的配置语法的命令：

```
httpd.exe -t 这个命令
```

其所在位置：apache 安装目录/bin/

php 的配置

首先，php 的配置，依赖于一个前提：

让 apache 知道 php.ini 文件在哪里！

在 apache 的配置文件 httpd.conf 中，添加一行：

```
PHPIniDir "php 语言包所在目录/"
```

时区配置：

```
data.timezone = PRC
```

php 的模块配置：

1) 指定模块所在位置：extension_dir = "php 语言包所在目录/ext"

2) 打开所需要的模块：extension = php_mysql.dll

mysql 的安装：

主机（站点）配置

一个站点的 2 个核心信息为：

```
ServerName 站点名称/服务器名/域名
```

```
DocumentRoot "站点的真实目录"
```

单站点配置项

端口监听：

```
Listen 端口号
```

设置主机（站点）的名字：服务器名，站点名

设置主机（站点）的物理路径：

目录访问权限

```
<Directory "要设置权限的物理路径">
```

```
Options Indexes #设置目录可显示“文件列表”
```

```
Order Deny,Allow #先拒绝后允许或先允许后拒绝，不管哪个顺序，都是后者覆盖前者
```

```
Allow from All
```

```
#下一行，可以让我们在文件夹中设定权限
```

```
AllowOverride all
```

```
</Directory>
```

目录访问权限中的“分布式权限”：

一个站点的任何一个文件夹，都可以对其进行“单独权限设置”：

1，需要在该文件夹中放入一个特殊名字的文件：.htaccess

2，该文件夹的内容，几乎可以跟 Directory 中的设置一样

3，该文件中的设置，优先于 Directory 中的设置，即如果有同样设置项但值不同，以.htaccess 中的为准；

4，.htaccess 中的设置项，无需重启 apache，就可以立即生效。

多站点的配置：

分 3 步：

1，在 httpd.conf 中，打开“虚拟主机配置文件”：apache 安装位置/ conf/ extra/ httpd-vhosts.conf

```
505 # Virtual hosts
506 Include conf/extra/httpd-vhosts.conf
507
```

2，在该虚拟主机配置文件（httpd-vhosts.conf）中，添加一行表示要进行多站点配置的 ip 端口：

```
NameVirtualHost *:80
```

3，然后，在其中，一个站点一个站点进行配置，每个站点的基本形式如下：

```
<VirtualHost>
    ServerName ....
    DocumentRoot .....
    <Directory .....>

    </Directory>
</VirtualHost>
```

站点别名的设置：

```
ServerAlias 别名 1 别名 2 ...
```

注意：它是写在 ServerName 并列的位置。

目录别名（虚拟目录）的设置：

就是设置一个“虚拟的目录名字”，以供外界访问“站点之外的内容”。

php 基本语法形式

php 的标记符

有以下形式：

形式 1（推荐）：

```
<?php
    这里要写符合 php 语法的语句
?>
```

形式 2：

```
<script language="php">
    这里要写符合 php 语法的语句
</script>
```

形式 3（不推荐）：

```
<?
```

这里要写符合 php 语法的语句

?>

它依赖于 php.ini 中的一个设置项:

```
225 ; http://php.net/short-open-tag
226 short_open_tag = Off
227
```

将其改为 On, 则上述形式就可以用了。

```
225 ; http://php.net/short-open-tag
226 short_open_tag = On
227
```

php 的结束标记 (比如 ?>), 在如下情况可以省略:

php 的语句之后, 再没有了 html 代码部分

php 的区分大小写特性

- 1, 变量区分大小写;
- 2, 常量通常默认也区分, 但可以人为设定为不区分 (但这种做法不推荐)
- 3, 其他场合的关键字都不区分, 比如函数名, 系统关键字 (for, if, return....)

一条语句使用一个分号结束

- 1, 在一个 php 的语句标记中的最后一个分号可省略

```
12 <body>
13 <?php
14 echo "<br />abc11";
15 echo "<br />abc12";
16 echo "<br />abc13"
17 ?>
18 <hr />
```

不能省略

可以省略

- 2, php 结束标记省略则不能省略最后一个分号

```
27 </html>
28 <?php
29 echo "<br />这是最后的php代码, 其后没有其他代码了";
30 echo "<br />这是最后的php代码, 其后没有其他代码了";
31 echo "<br />这是最后的php代码, 其后没有其他代码了"
32
```

但: 这里不能省略, 因为其后没有 php 的结尾标记

php 的注释

单行注释：

形式 1: //注释内容

形式 2: #注释内容

多行注释：

/* 注释内容，可以多行 */

2 个多行注释的技巧：

有时候，我们因为测试或别的原因，需要将一大段代码多次进行“注释”或“反注释”；，则此时，可以使用如下 2 个技巧来方便实现：

技巧 1：

```
14  /*
15  echo "<br />代码1";      全部注释了！
16  echo "<br />代码2";
17  echo "<br />代码3,代表多行需要注释的代码";
18  /**/
```

反注释：

```
14  /**/
15  echo "<br />代码1";      这样就有全部取消注释了
16  echo "<br />代码2";
17  echo "<br />代码3,代表多行需要注释的代码";
18  /**/
```

技巧 2：

```
17
▶ 20  if( 1 == 0 ){          注释掉了
21  »  echo "<br />代码1";
22  »  echo "<br />代码2";
23  »  echo "<br />代码3,代表多行需要注释的代码";
24  }
```

反注释：

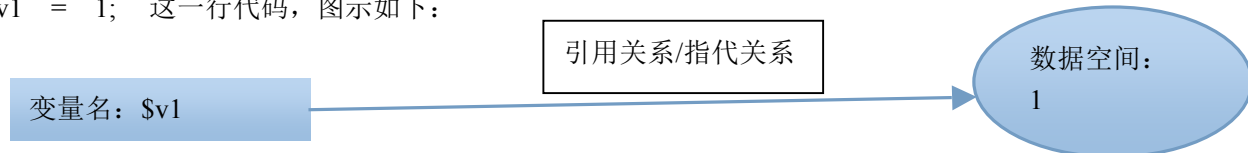
```
20  if( 1 == 1 ){
21  »  echo "<br />代码1";
22  »  echo "<br />代码2";
23  »  echo "<br />代码3,代表多行需要注释的代码";
24  }
```

变量

基本理解

变量可以理解为只是一个代表一定存储空间及其中的数据的一个“标识符”——也就是一个名字。

`$v1 = 1;` 这一行代码，图示如下：



这表明，使用该变量，就是在使用该数据空间的数据值！比如：

```
echo $v1;           //输出的是数据 1
```

```
$v2 = $v1 + 3;      //此时，其实进行的是 1+3 的计算！
```

使用形式：每个变量名前面必须以\$开头。

定义形式：php 中，不支持“单纯定义”一个变量，而是，在第一次一个变量赋值的时候，就算是定义变量！

变量的基本操作

只有 4 个操作：

赋值：

等号（=）的左边放变量名，右边放“数据”，就是赋值；

```
$v1 = 1;
```

取值：

任何需要一个数据的语句中使用一个变量，此时就是指从该变量取得该变量的值——取值；

概括来说，大约有如下情形：

```
echo $v1;           //取出 v1 的值并输出
```

```
$v2 = $v1;          //取出 v1 的值并给其他变量赋值；
```

```
$v2 = $v1 + 3;      //取出 v1 的值并与 3 进行运算！
```

```
$v2 = round($v1);   //取出 v1 的值并并使用函数 round()对其进行四舍五入运算
```

判断 isset(变量名):

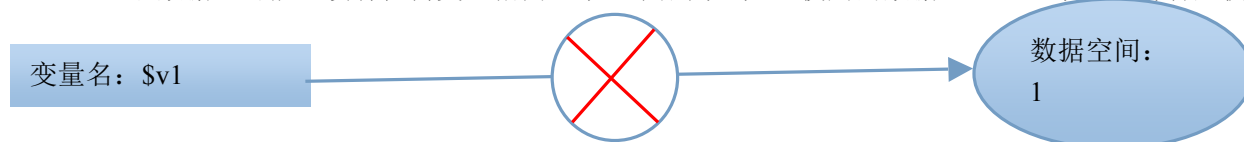
作用：判断该变量是否存在，或该变量是否有数据值！存在或有数据值，就是 true，否则就是 false

```
14 $v1 = isset($s1);» //判断$s1是否存在，结果是false
15 $s2 = 2;
16 $v2 = isset($s2);» //true
17
18 $s3 = false;
19 $v3 = isset($s3);» //true
20
21 $s4 = "";
22 $v4 = isset($s4);» //true
23
24 $s5 = null;»//特别注意：null这个“值”（数据）的含义就是“没有数据”
25 $v5 = isset($s5);» //false;
```

删除 unset(变量名):

含义：删除一个变量，并不是指将该变量从程序中删掉，而是，“断开”该变量名跟该变量原有的数据值之间的“引用关系”（联系）！，此时，会有这样的结果：

- 1，该变量名已经不指向（引用）任何数据了，则此时其 isset()判断的结果就是 false
- 2，该数据（可能）没有任何变量指向它了，就成为“无法使用的数据”——通常就被自动回收了



```
$v1 = 1;
unset($v1);
$s1 = isset($v1);      //false
```

变量命名规则

基本规则——保证程序的正确性

- 以字母或下划线开头
- 后跟任意数量（含 0 个）的字母，数字和下划线

行业规则——保证程序的可读性

有 3 种常见的命名法：

1，骆驼命名法：首单词小写，其后每个单词首字母大写

举例：\$name \$myName \$myFatherName

2，帕斯卡命名法：每个单词首字母大写

举例：\$Name \$MyName \$MyFatherName

3，下划线分割法：每个单词小写，并且之间用下划线分开

举例：\$name \$my_name \$my_father_name

变量的传值方式

1，变量的传值方式，是指“一个变量，传给另一个变量”的内部细节形式——单对单；

2，变量的传值方式，只有 2 中：值传递，引用传递；

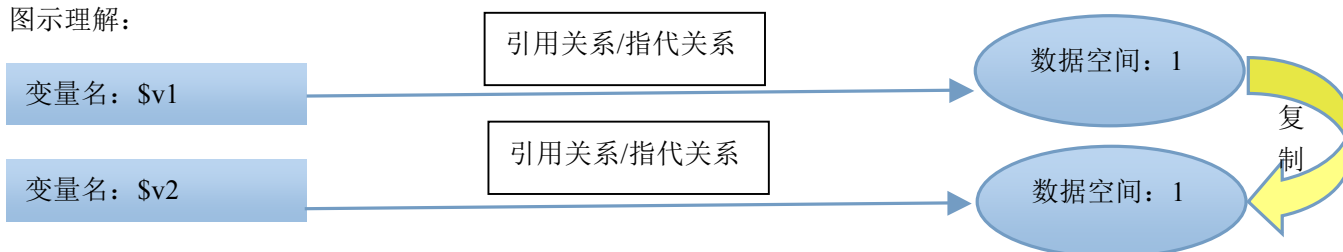
值传递

```
$v1 = 1;
```

```
$v2 = $v1; //这就是值传递
```

简单理解：将\$v1 的值取出来（注意：\$v1 中的值还在），然后再用该值给\$v2 赋值。

图示理解：



可见，值传递，就是变量 v1 的值进行了复制，然后在给另一个变量 v2 赋值。

注意：

1 这两个变量此时是值相等的；

2 这两个变量又是互相独立的——互不影响；

即 \$v1 = 10; 则 echo \$v2 ; //输出 1

引用传递

php 中，只有一种语法形式可以实现变量的引用传值方式：&符号。

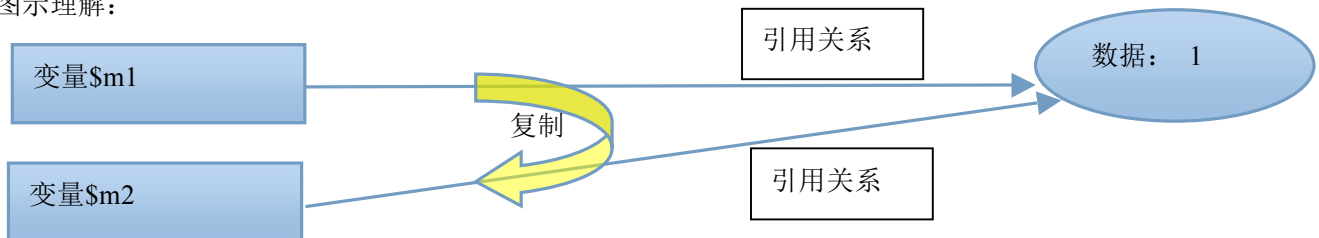
举例如下：

```
$m1 = 1;
```

`$m2 = & $m1;` //引用传值方式

简单理解： 将变量\$m1 跟其数据值之间的“引用关系”，复制一份，再给予变量\$m2，即，此时，变量\$m2 也同样具有跟原来数据的一个“引用关系”（指向关系）；

图示理解：



结果是：

- 1，此时仍然有了 2 个变量，但只有一个数据值（数据空间），2 个变量都共同指向该数据空间。
- 2，对其中任何一个变量的操作，其实都是在操作数据值（空间）；

代码对比演示：

```
14 $v1 = 1;
15 $v2 = $v1;» //值传递;
16 $v1 = 10;
17 echo "<br />v1 = $v1";//10
18 echo "<br />v2 = $v2";//1
19 unset($v1);»» //断开$v1跟其数据的关系
20 $r1 = isset($v2);» //??
21 echo "<br />"; var_dump($r1);//true
22
23 echo "<hr />";
24
25 $m1 = 1;
26 $m2 = & $m1;» » //引用传值方式
27 $m1 = 10;
28 echo "<br />m1 = $m1";//10
29 echo "<br />m2 = $m2";//10
30 unset($m1);»» //断开$m1跟其数据的关系
31 $r2 = isset($m2);» //??
32 echo "<br />"; var_dump($r2);//true
33
```

可变变量：

`$s1 = "abc";` //这是一个变量，里面存储的是字符串"abc"

`$abc = 10;` //

`echo $$s1;` //???, 输出 10

理解：

- 1, 在 php 中, 一个"\$" 后面, 总是跟着一个变量名!
- 2, 这里, echo 输出的这个变量(以第一个\$为标识) 的名字是: \$s1, 即"abc"
- 3, 所以, 这里输出的是 \$abc, 即 10
- 4, 这种连续出现 "\$" 的变量形式, 就是所谓的 "可变变量";

```
14 $s1 = 'abc';» » //这是一个变量, 里面存储的是字符串"abc"
15 $abc = 10;» » //
16 echo $$s1;» » ///???, 输出10
17
```

10

```
18 //以下演示"可变变量"的灵活性:
19 //所谓可变变量, 其实就是变量的名字是可以"动态变化"以获取不同的数据值
20 $v1 = 1;
21 $v2 = 12;
22 $v3 = 33;
23 $v4 = 44;
24 $v5 = 115;
25 //求这5个变量的和;
26 //其他语言, 只能"一个一个"加起来。
27 $sum = 0;» //用于存储总和
28 for($i = 1; $i <= 5; $i++){
29 » $v = "v" . $i;» //这里, 结果其实只是一个"字符串", 比如"v1","v2"...
30 » $sum += $$v;
31 }
32 echo "<br />sum = $sum";
```

sum = 205

预定义变量

所谓预定义变量, 其实指, php 这个语言工具中, 预先就定义好的变量;
我们只是“拿来使用”。

综述

- 主要有: \$_GET, \$_POST, \$_REQUEST, \$_SERVER, \$GLOBALS ,
- 均是数组
- 系统定义与维护——即我们不应该其给其赋值或销毁其值, 只应该去“用其值”。
- 具有超全局作用域——哪里都可以使用。
- 不同情形下可能具有不同的值

\$_POST 变量

含义：

它代表用户通过表单以 post 方式（ method="post" ）提交的时候所提交的所有数据——这个称为 post 数据。

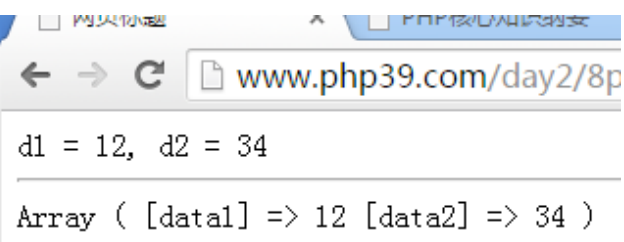
基本演示：

7form_post.html 页面：

```
13 <!--  
14 以下表单的数据，点击提交后，会将所有表单数据提交给  
15 8post_data.php这个页面（文件），并在该文件中去处理  
16 其实就是指：程序进入该文件中运行。  
17 -->  
18 <form action="8post_data.php" method="post">  
19 » 数据1: <input type="text" name="data1" />  
20 » <br />  
21 » 数据2: <input type="text" name="data2" />  
22 » <br />  
23 » <input type="submit" value="提交" />  
24 </form>
```

8post_data.php 页面及输出结果为：

```
13 <?php  
14 //先一个一个获取数据：  
15 $d1 = $_POST['data1'];  
16 $d2 = $_POST['data2'];  
17 //然后输出，看看：  
18 echo "d1 = $d1, d2 = $d2";  
19  
20 echo "<hr />";  
21 //再来看看整个$_POST这个数组数据！  
22 print_r($_POST);  
23 ?>
```



isset(变量): 判断变量是否存在，或变量是否为空（null）；如果存在，就是 true，否则就是 false

empty(变量): 判断变量的“内容”是否为空的（不是 null 的空，而是没有内容），基本上，是一些硬性规定，如下数据都是“空的”： 0, “”, “0”, false, null, array()空数组也是空

如果一个变量内容是空的，empty()返回的结果是 true，否则是 false

小案例：php 页面计算器：

```

46  <form action="" method="post">
47  »   <input type="text" name="data1" value="<?php echo $num1; ?>" />
48  »   <select name="yunsuanfu" >
49  »       <option value="+" <?php if($fuhao==""){ echo 'selected="selected"'; } ?> >+</option>
50  »       <option value="-" <?php if($fuhao=="-"){ echo 'selected="selected"'; } ?> >-</option>
51  »       <option value="*" <?php if($fuhao=="*"){ echo 'selected="selected"'; } ?> >*</option>
52  »       <option value="/" <?php if($fuhao=="/"){ echo 'selected="selected"'; } ?> >/</option>
53  »   </select>
54  »   <input type="text" name="data2" value="<?php echo $num2; ?>" />
55  »   <input type="submit" value="=" />
56  »   <input type="text" name="result" value="<?php echo $jieguo; ?>" />
57  </form>
58

```

```

13  <?php
14  if(!empty($_POST)){
15  »   $num1 = $_POST['data1'];
16  »   $num2 = $_POST['data2'];
17
18  »   $fuhao = $_POST['yunsuanfu'];
19  »   //echo "符号为: $fuhao";
20  »
21  »   if( $fuhao == '+' ){
22  »       »   $jieguo = $num1 + $num2;
23  »       »   }
24  »   else if( $fuhao == '-' ){
25  »       »   $jieguo = $num1 - $num2;
26  »       »   }
27  »   else if( $fuhao == '*' ){
28  »       »   $jieguo = $num1 * $num2;
29  »       »   }
30  »   else if( $fuhao == '/' ){
31  »       »   $jieguo = $num1 / $num2;
32  »       »   }
33  »   }
34  } else{
35  »   //当第一次进来是（没有post数据），
36  »   //设定这4个变量，保证后面的变量使用不会出错
37  »   $jieguo = "";
38  »   $num1 = "";
39  »   $num2 = "";
40  »   $fuhao = "";
41  »   }
42  ?>

```

\$_GET 变量

含义:

它代表用户通过 get 方式（有 5 种 get 形式）提交的时候所提交的所有数据——这个称为 get 数据。
小提示：get, post 没有翻译！

提交 get 数据有 5 种形式：

形式 1：

```
<form action="目标文件.php" method="get">
    <input type="text" name="data1" />
    <input type="text" name="data2" />
    <input type="submit" value="提交" />
</form>
```

这种形式的 get 数据，跟 post 数据类似，数据内容由用户填写或选择而得到！

形式 2：

```
<a href="目标文件.php? data1=5&data2=cctv&age=18">链接文字</a>
```

说明：

- 1，它只是一个链接而已，只是在链接文件名的后面加上“？”，然后一个一个“串接数据”；
- 2，数据形式为：数据项名称=数据值，相互之间用“&”符号隔开
- 3，这种形式的数据也同样是“点击链接”就提交的 get 数据，但用户只能选择点还是不点，而不能修改数据。

形式 3：

```
<script>
    location.href = "目标文件.php? data1=5&data2=cctv&age=18";
</script>
```

说明：

- 1，该语句可以看做是通过 js 技术实现的页面跳转功能，跟 a 标签的连接功能完全一样！
- 2，其中该语句，通常都是放在一个函数中，然后因为某个事件发生而去调用该函数！

形式 4：

```
<script>
    location.assign("目标文件.php? data1=5&data2=cctv&age=18");
</script>
```

说明：

该语句其实跟形式 3 完全一样功能，只是 location 这个对象实现页面跳转的另一个语法形式而已！

形式 5：php 的跳转语法

```
<?php
    //语法形式： header("location: 目标网页地址");

    header("location: 目标文件.php? data1=5&data2=cctv&age=18");
?>
```

小结:

上述多种 get 形式提交数据，都要理解为:

跳转到某个页面，并“同时”携带（提交）一定的 get 数据过去！

不管哪种形式的 get 数据提交，接收 get 数据，都只有一种形式，跟 post 类似:

`$v1 = $_GET['数据项名称'];` //取得一个 get 数据项的值；比如`$_GET['data1']`，`$_GET["age"]`;

也可以“输出”所有 get 数据:

`print_r($_GET);` 或 `var_dump($_GET);`

\$_REQUEST 变量

含义:

一句话，它是`$_GET` 变量和`$_POST` 变量数据的“合集”：即，它里面同时存储了这两种数据。

```
27 <form action="" method="post">
28 » 数据1: <input type="text" name="data1" />
29 » <br />
30 » 数据2: <input type="text" name="data2" />
31 » <br />
32 » <input type="submit" value="提交" />
33 </form>
```

数据1: 11
数据2: 22
提交

结果为:

```
14 if(!empty($_POST)){
15 » echo "<p>post数据: <br />";
16 » print_r($_POST);
17 }
18 if(!empty($_GET)){
19 » echo "<p>get数据: <br />";
20 » print_r($_GET);
21 }
22 if(!empty($_REQUEST)){
23 » echo "<p>request数据: <br />";
24 » print_r($_REQUEST);
25 }
```

post数据:
Array ([data1] => 11 [data2] => 22)
request数据:
Array ([data1] => 11 [data2] => 22)

如果表单为 get 提交方式:

```
27 <form action="" method="get">
28 » 数据1: <input type="text" name="data1" />
29 » <br />
30 » 数据2: <input type="text" name="data2" />
31 » <br />
32 » <input type="submit" value="提交" />
33 </form>
```

数据1: 11
数据2: 22
提交

结果为:

```

14  if(!empty($_POST)){
15      »   echo "<p>post数据: <br />";
16      »   print_r($_POST);
17  }
18  if(!empty($_GET)){
19      »   echo "<p>get数据: <br />";
20      »   print_r($_GET);
21  }
22  if(!empty($_REQUEST)){
23      »   echo "<p>request数据: <br />";
24      »   print_r($_REQUEST);
25  }

```



怎么能同时具有 get 和 post 数据呢？

只有一个方式：

```

<!--以下 action 中的 ? 号后面的数据都会以 get 方式提交-->
<form   action="目标文件.php? data1=5&data2=cctv&age=18"   method="post">
<!--以下表单项的数据都会以 post 方式提交-->
<input type="text"   name="n1" />
<input type="text"   name="n1" />
<input type="submit" value="提交" />
</form>

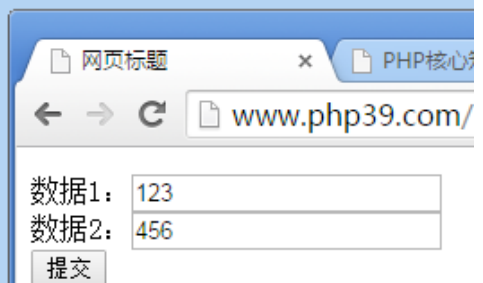
```

举例：

```

27  <form   action="10request.php?d1=5&d2=cctv&age=18" method="post">
28      »   数据1: <input type="text" name="data1" />
29      »   <br />
30      »   数据2: <input type="text" name="data2" />
31      »   <br />
32      »   <input type="submit" value="提交" />
33  </form>
34  </body>
35  </html>

```

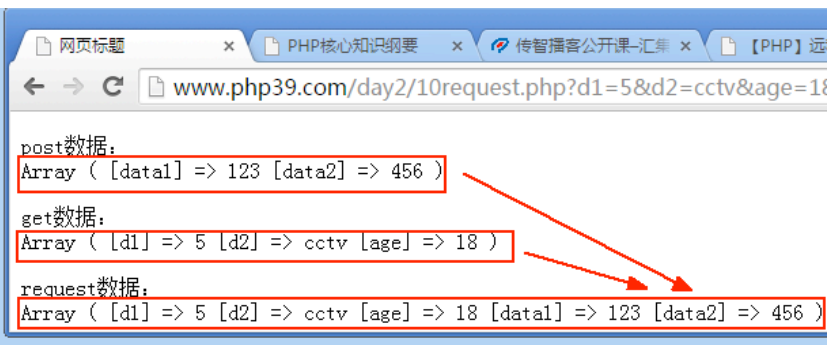


提交后结果为：

```

14  if(!empty($_POST)){
15      »   echo "<p>post数据: <br />";
16      »   print_r($_POST);
17  }
18  if(!empty($_GET)){
19      »   echo "<p>get数据: <br />";
20      »   print_r($_GET);
21  }
22  if(!empty($_REQUEST)){
23      »   echo "<p>request数据: <br />";
24      »   print_r($_REQUEST);
25  }

```



request 数据取值时，跟 get 数据和 post 也完全一样！

`$_REQUEST['数据项名称'];`


```

» echo "<br />也可以这样写：";
» echo "<br />" . $_REQUEST['d1'];
» echo "<br />" . $_REQUEST['d2'];
» echo "<br />" . $_REQUEST['age'];
» echo "<br />" . $_REQUEST['data1'];
» echo "<br />" . $_REQUEST['data2'];

```

其实是get数据

其实是post数据

当 post 数据和 get 数据的数据项名称相同时（其实我们反对这么做），默认是 post 数据覆盖了 get 数据。

不过这个状况同样可以在 php.ini 中设置：

默认时：

```

693 ; http://php.net/request-order
694 request_order = "GP"
695

```

就是GET和POST，后者覆盖前者。

可修改为：

```

693 ; http://php.net/request-order
694 request_order = "PG"
695

```

此时就是GET数据覆盖POST数据

\$_SERVER 变量

含义：

它代表在一次浏览网页的过程中的浏览器端的一些信息或服务器端的一些信息。

我们只是在程序中可以拿到这些信息，并用于编程所需！比如：取得用户的访问 IP 地址。

注意：

这种信息，随着不同的页面，和不同的服务器，以及不同的时刻，都可能不同！

要求：

大约有 30 个左右的信息，我们只要知道其中 5 个左右！主要有：

\$_SERVER['REMOTE_ADDR']：获取访问者的 ip 地址

\$_SERVER['SERVER_ADDR']：获取服务器所在的 ip 地址

\$_SERVER['SERVER_NAME']：获取服务器的名字，其实就是站点设置中的 servername

\$_SERVER['DOCUMENT_ROOT']：获取站点的真实物理地址，其实就是站点设置中的 documentroot

\$_SERVER['PHP_SELF']：获取当前网页地址（不含域名部分）

\$_SERVER['SCRIPT_FILENAME']：获取当前网页地址物理路径

\$_SERVER['QUERY_STRING']获取当前网页地址中的所有 get 数据（就是？号后面部分），但只是一个整体的字符串而已。

```

14 echo "<pre>";
15 print_r($_SERVER);
16 echo "</pre>";
17
18 echo "<table border='1'>";
19 foreach($_SERVER as $key => $value){
20     » echo "<tr>";
21     » echo "<td>$key</td>";
22     » echo "<td>$value</td>";
23     » echo "</tr>";
24 }
25 echo "</table>";

```

\$GLOBALS 变量

含义：

它也是一个“重复性数据”，它里面存储了我们自己定义的所有“全局变量”。

举例：

`$v1 = 1;` //定义了一个全局变量，

此时，就有了这样一个数据：`$GLOBALS['v1']`，其值就是 1

`echo $v1;` //输出 1

`echo $GLOBALS['v1'];` //输出 1

这个变量，主要是用于在局部范围不可以使用全局变量的时候，又需要该全局变量的值，此时就可以用它来取得该全局变量的值。

```

14 $v1 = 1;
15 echo "<pre>";
16 print_r($GLOBALS);           [v1] => 1
17 echo "</pre>";                )

```

再定义一个新的变量：

```

19 $v2 = 2;
20 echo "<pre>";
21 print_r($GLOBALS);           [v1] => 1
22 echo "</pre>";                [v2] => 2

```