HTTP 协议

HTTP 协议初步认识

HTTP 协议概念

HTTP 协议，即超文本传输协议(Hypertext transfer protocol)。是一种详细规定了浏览器和万维网(WWW = World Wide Web)服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议。
HTTP 协议是用于从 WWW 服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP 协议特点

1） 客户/服务器模式：客户端（浏览器）/服务端
2） 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。
3） 灵活：HTTP 允许传输任意类型的数据对象（MIME 类型）
4）无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
5）无状态：HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

HTTP 协议分类

1）http 请求协议：浏览器向服务器发起请求的时候需要遵循的协议
2）http 响应协议：服务器向浏览器发起响应的时候需要遵循的协议

HTTP 请求

请求行

1） 形式：请求方式　资源路径　协议版本号
2） GET /index.php HTTP/1.1

最早的时候 HTTP 协议有过 1.0，请求行独占一行（第一行）

请求头

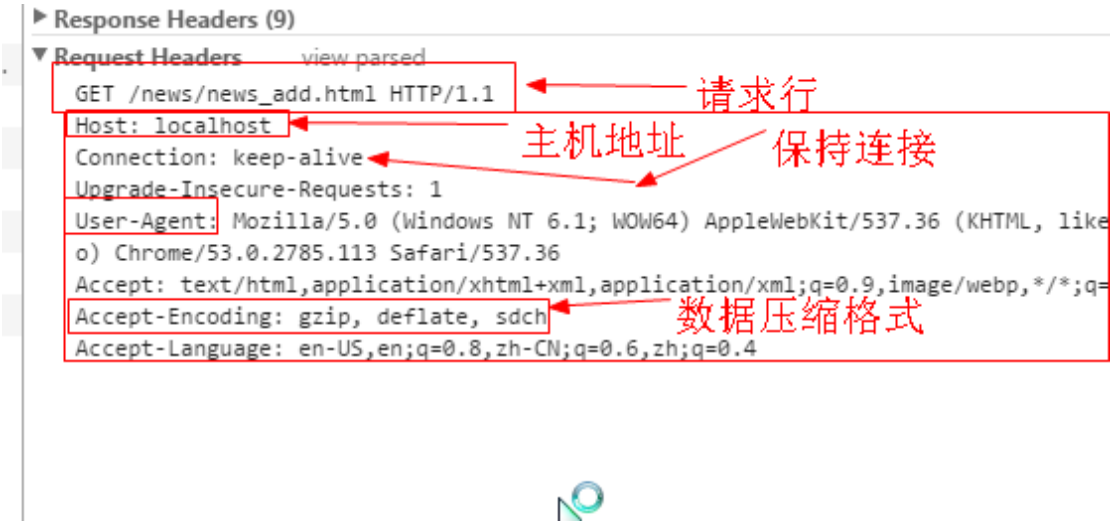
请求头就是各项协议内容：具体的协议内容不会每次都使用全部
1）Host：请求的主机地址（必须）
2）Accept：当前请求能够接收服务器返回的类型（MIME 类型）
3）Accept-Language：接收的语言
4）User-Agent：客户浏览器所在点的一些信息

请求头不固定数量，每个请求协议也是独占一行，最后会有一行空行（用来区分请求头和请求体）

请求体

请求数据：POST 请求会有请求体。GET 请求所有的数据都是跟在 URL 之后，会在请求行中的资源路径上体现。

基本格式：资源名字=资源值&资源名字=资源值…



HTTP 响应

响应行

1）形式：协议版本号 状态码 状态消息（独占一行）
HTTP/1.1 200 ok
2）200 ok： 成功
3）403 Forbidden： 没权限访问

4）404 Not Found： 未找到页面
5）500 Server Internal Error： 服务器内部错误

响应头

具体协议内容
1）时间：Wed, 16 Sep 2017 11:43:33 GMT
2）服务器：Server: Apache/2.2.22 (Win32) PHP/5.3.13
3）内容长度：Content-Length: 1571，数据具体的字节数（响应体）
4）内容类型：Content-Type: text/html：告诉浏览器对应的数据格式

列举了几个常见的响应头，并不是全部：响应头一个占一行，最后一行空行（区分响应头和响应体）

响应体

实际服务器响应给浏览器的内容

常用 HTTP 状态码

状态码 200：成功
状态码 403：forbidden，拒绝访问（没有权限）
状态码 404：NOT FOUND，找不到
状态码 500：服务器问题

服务器正在处理过程中

1xx: 信息

消息:	描述:
100 Continue	服务器仅接收到部分请求，但是一旦服务器并没有拒绝该请求，客户端应该继续发送其余的请求。
101 Switching Protocols	服务器转换协议：服务器将遵从客户的请求转换到另外一种协议。

服务器正常且正确处理

2xx: 成功

消息:	描述:
200 OK	请求成功（其后是对GET和POST请求的应答文档。）
201 Created	请求被创建完成，同时新的资源被创建。
202 Accepted	供处理的请求已被接受，但是处理未完成。
203 Non-authoritative Information	文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝。
204 No Content	没有新文档。浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。
205 Reset Content	没有新文档。但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容。
206 Partial Content	客户发送了一个带有Range头的GET请求，服务器完成了它。

请求的目标已经转移或者需要更新

3xx: 重定向

消息:	描述:
300 Multiple Choices	多重选择。链接列表。用户可以选择某链接到达目的地。最多允许五个地址。
301 Moved Permanently	所请求的页面已经转移至新的url。
302 Found	所请求的页面已经临时转移至新的url。
303 See Other	所请求的页面可在别的url下被找到。
304 Not Modified	未按预期修改文档。客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。
305 Use Proxy	客户请求的文档应该通过Location头所指明的代理服务器提取。
306 *Unused*	此代码被用于前一版本。目前已不再使用，但是代码依然被保留。
307 Temporary Redirect	被请求的页面已经临时移至新的url。

客户端出错了

4xx: 客户端错误

消息:	描述:
400 Bad Request	服务器未能理解请求。
401 Unauthorized	被请求的页面需要用户名和密码。
402 Payment Required	此代码尚无法使用。
403 Forbidden	对被请求页面的访问被禁止。
404 Not Found	服务器无法找到被请求的页面。
405 Method Not Allowed	请求中指定的方法不被允许。
406 Not Acceptable	服务器生成的响应无法被客户端所接受。
407 Proxy Authentication Required	用户必须首先使用代理服务器进行验证，这样请求才会被处理。
408 Request Timeout	请求超出了服务器的等待时间。
409 Conflict	由于冲突，请求无法被完成。
410 Gone	被请求的页面不可用。
411 Length Required	"Content-Length" 未被定义。如果无此内容，服务器不会接受请求。
412 Precondition Failed	请求中的前提条件被服务器评估为失败。
413 Request Entity Too Large	由于所请求的实体的太大，服务器不会接受请求。
414 Request-url Too Long	由于url太长，服务器不会接受请求。当post请求被转换为带有很长的查询信息的get请求时，就会发生这种情况。
415 Unsupported Media Type	由于媒介类型不被支持，服务器不会接受请求。
416	服务器不能满足客户在请求中指定的Range头。

服务器出错

5xx: 服务器错误

消息:	描述:
500 Internal Server Error	请求未完成。服务器遇到不可预知的情况。
501 Not Implemented	请求未完成。服务器不支持所请求的功能。
502 Bad Gateway	请求未完成。服务器从上游服务器收到一个无效的响应。
503 Service Unavailable	请求未完成。服务器临时过载或当机。
504 Gateway Timeout	网关超时。
505 HTTP Version Not Supported	服务器不支持请求中指明的HTTP协议版本。

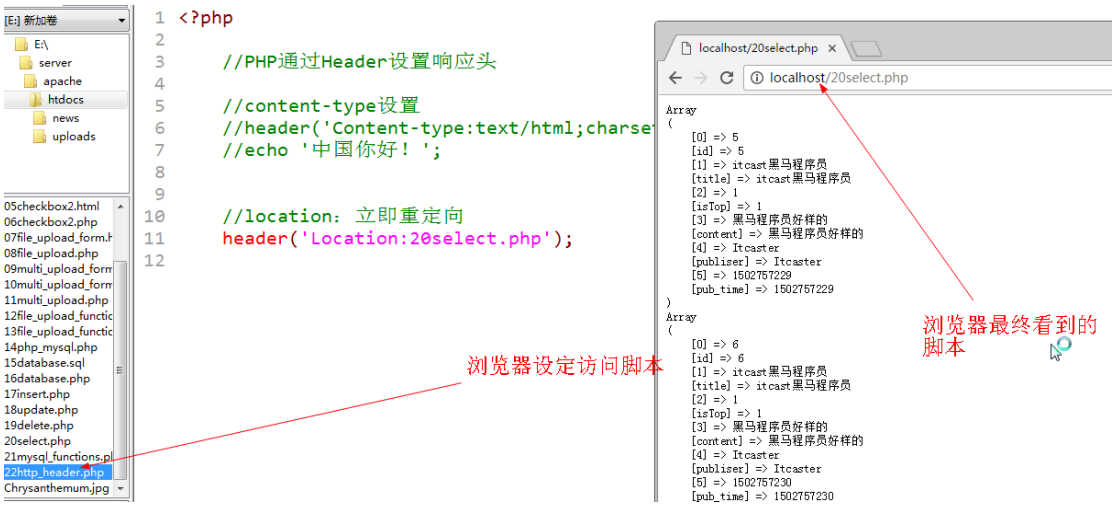
HTTP 响应

常见 HTTP 响应设置及使用

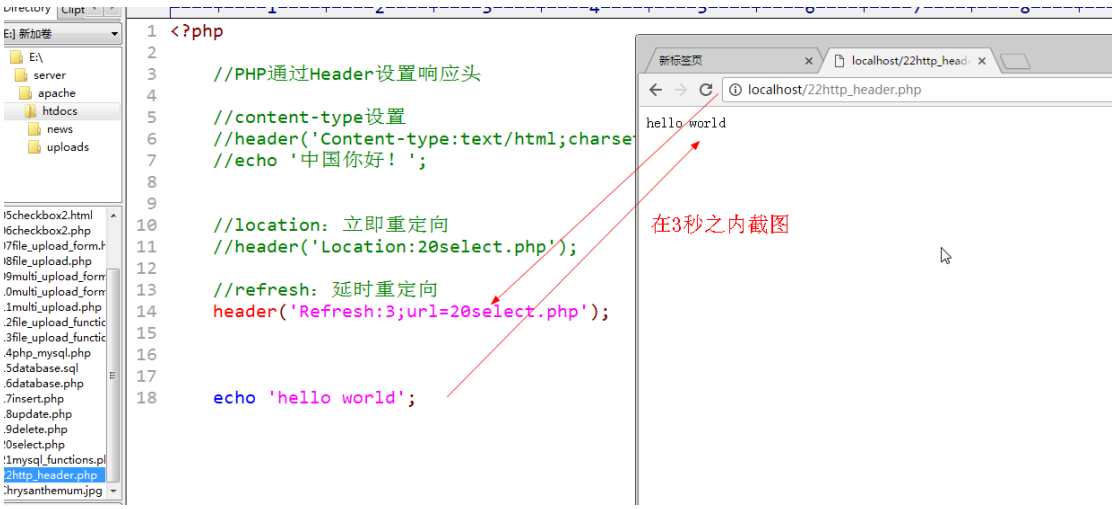
PHP 中针对 HTTP 协议（响应）进行了底层设计，可以通过函数 header 来实现修改 HTTP 响应（响应头）

注意事项：
1、 Header 可以设计 HTTP 响应，因为 HTTP 协议特点是：响应行，响应头（空行结尾），响应体。认为通过 header 设计响应头的时候，不应该有任何内容输出，所以一旦产生内容输出（哪怕一个空格），系统都会认为响应头已经结束而响应体开始了，所有如果先输出内容后设置响应头（header 使用），理论设置无效；
2、 在 PHP5 以后，增加程序缓存内容：允许服务器脚本在输出内容的时候，不直接返回浏览器而是现在服务器端使用程序缓存保留（php.ini 中使用 output_buffering），有了该内容之后，在程序缓存内会自动调整响应头和响应体（允许响应头在已经输出的内容之后再设置），但是此时会报错（警告）。
总结：header 设置响应体之前不要有任何输出

Location：重定向，立即跳转（响应体不用解析）
浏览器在解析服务器响应的时候：先判定响应行，继续响应头，最后响应体：location 是在响应头中，所以浏览器一旦见到该协议项，不再向下解析。

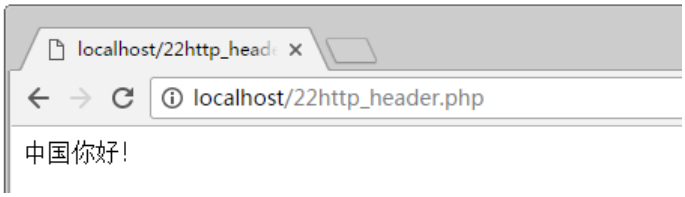


Refresh：重定向，定时跳转（响应体会解析）
延时重定向：浏览器会根据具体时间延迟后在访问指定跳转链接：浏览器在准备跳转访问之前，会继续解析 HTTP 协议（响应头和响应体）



Content-type：内容类型，MIME 类型
通过内容告知（MIME 类型），浏览器正确解析内容

```php
//content-type设置
header('Content-type:text/html;charset=utf-8');
echo '中国你好！ ';
```



Content-disposition：内容类型，MIME 类型扩展，激活浏览器文件下载对话框
浏览器在解析内容的时候，默认是直接解析：那么有时候需要浏览器不解析，当做内容下载成文件

```php
//refresh: 延时重定向
//header('Refresh:3;url=20select.php');

//Content-disposition
header('Content-disposition:attachment;filename=girl.jpg');

echo 'hello world';
```

PHP 模拟 HTTP 请求

原理

PHP 可以通过模拟 HTTP 协议发起 HTTP 请求
CURL 是一个非常强大的开源库，支持很多协议，包括 HTTP、FTP、TELNET 等，我们使用它来发送 HTTP 请求。它给我 们带来的好处是可以通过灵活的选项设置不同的 HTTP 协议参数，并且支持 HTTPS。CURL 可以根据 URL 前缀是“HTTP” 还是“HTTPS”自动选择是

否加密发送内容。

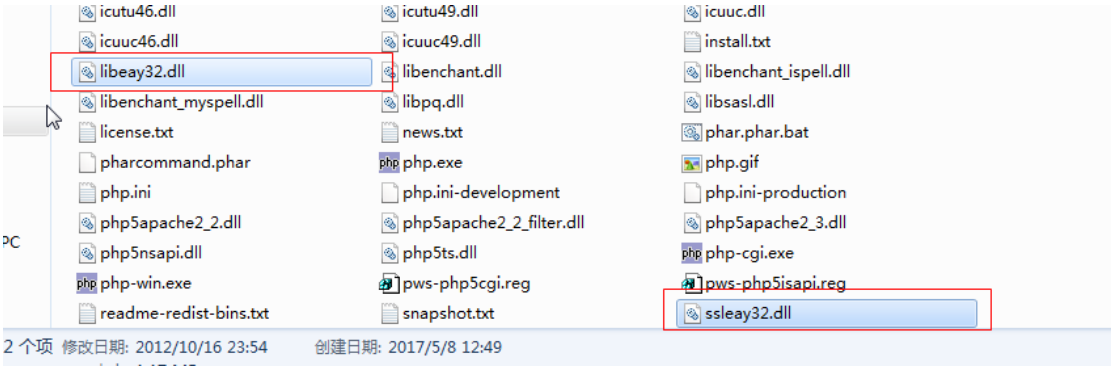
前提条件：HTTP 协议的客户端/服务端模式，HTTP 协议不局限于一定要浏览器访问

Curl 扩展库使用

1、 开启 CURL 扩展

```
368 ; Be sure to appropriately set the extension_d:
369 ;
370 ;extension=php_bz2.dll
371 extension=php_curl.dll
372 ;extension=php_fileinfo.dll
373 ;extension=php_gd2.dll
```

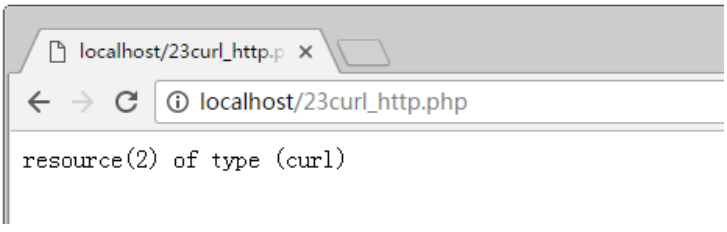
2、 有的时候 PHP 版本会出现即使开启了扩展也无法使用扩展的可能：因为 CURL 找不到对应的 dll 文件：需要将相关的 DLL 文件放到 C:windows



3、 重启 Apache 应用

1）建立连接：curl_init()：激活一个 CURL 连接功能

```
//开启会话
$ch = curl_init();
var_dump($ch);
```



2）设置请求选项：curl_setOpt()：设定选项

CURLOPT_URL：连接对象

CURLOPT_RETURNTRANSFER:将服务器执行的结果(响应)以文件流的形式返回给请求界面(PHP 脚本)

CURLOPT_POST：是否才有 POST 方式发起请求（默认请求是 GET）

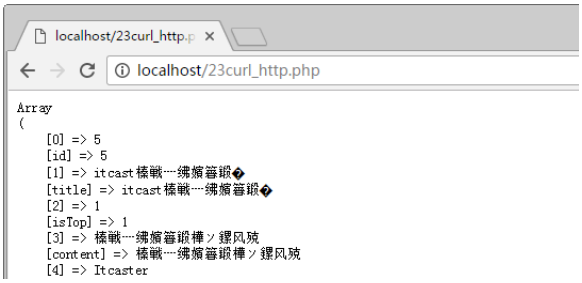
CURLOPT_POSTFIELDS：用来传递 POST 提交的数据，分为两种方式：字符串（name=abc&password=123）以及数组形式（array(‘name’=>’ abc’,…)）

CURLOPT_HEADER：是否得到响应的 header 信息（响应头），默认不获取

```
//var_dump($ch);

//设置连接选项
curl_setopt($ch,CURLOPT_URL,'localhost/20select.php');  //连接选项
curl_setopt($ch,CURLOPT_RETURNTRANSFER,TRUE);           //文件流形式返回数据（不直接显示）
curl_setopt($ch,CURLOPT_HEADER,0);                          //是否获取响应头信息
```

3）执行请求：curl_exec()：执行选项（与服务器发起请求），得到服务器返回的内容

```
curl_setopt($ch,CURLOPT_HEADER,0);

//执行
$content = curl_exec($ch);
echo $content;
```



乱码原因：20select.php 中的内容有告知浏览器，但是当前是被服务器脚本 23curl_http.php 访问的，没有做解析；输出给浏览器之后，需要当前 23curl_http.php 告知浏览器对应的字符集

4）关闭连接：curl_close()：关闭资源

```
//关闭资源
curl_close($ch);
```

文件编程

文件编程初步认识

文件编程的必要性

文件编程指利用 PHP 代码针对文件（文件夹）进行增删改查操作。

在实际开发项目中，会有很多内容（文件上传、配置文件等）具有很多不确定性，不能在一开始就手动的创建，需要根据实际需求和数据本身来进行管理，这个时候就可以使用 PHP 文件编程来实现代码批量控制和其他操作。

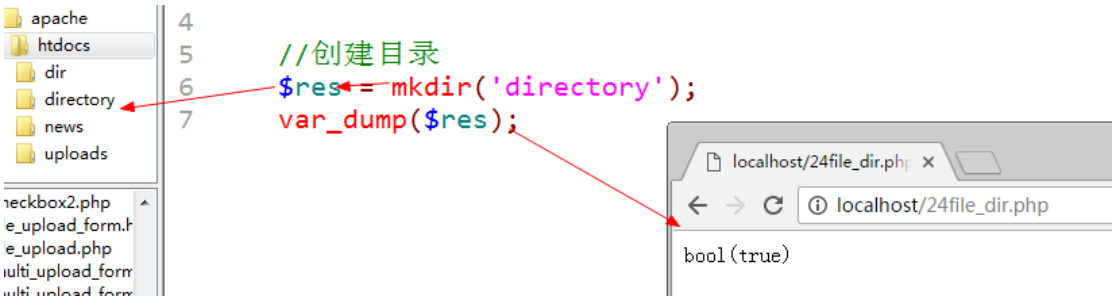
文件操作的分类

1） 目录操作：文件夹，用来存放文件的特殊文件
2） 文件操作：用来存放内容

目录操作

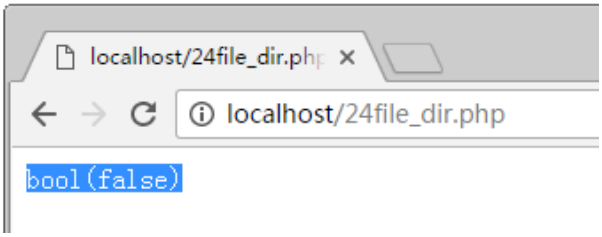
文件操作创建目录结构

1）mkDir(路径名字)：创建成功返回 true，创建失败返回 false



有些操作为的就是得到一个想要的结果，如果结果本身就存在，那么可以忽略得到过程的错误：抑制错误

```
//创建目录
$res = @mkdir('directory');
var_dump($res);
```



删除目录

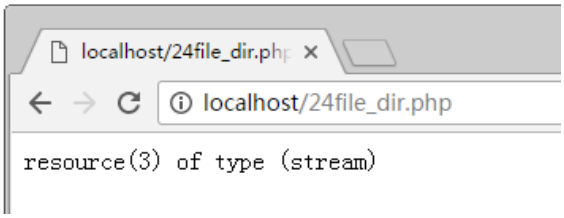
1）rmDir(指定文件夹路径)：移出文件夹

读取目录

读取方式：将文件夹（路径）按照资源方式打开

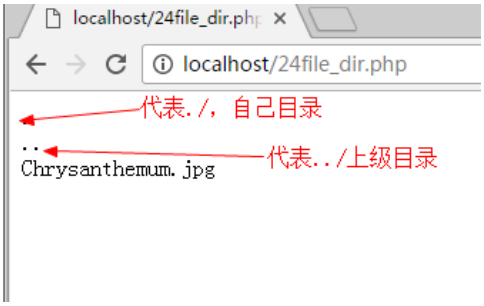
1）openDir()：打开资源，返回一个路径资源，包含指定目录下的所有文件（文件夹）

```
//读取目录
$r = opendir('uploads');
var_dump($r);
```



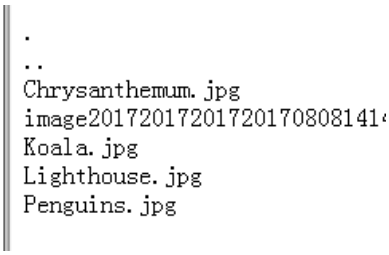
2）readDir()：从资源中读取指针所在位置的文件名字，然后指针下移，直到指针移出资源

```
$r = opendir('uploads');
//var_dump($r);

//读取资源
echo readdir($r),'<br/>';
echo readdir($r),'<br/>';
echo readdir($r),'<br/>';
```



读取所有内容：遍历操作

```
//循环遍历输出
while($file = readdir($r)){
    echo $file,'<br/>';
}
```



关闭目录

1）closeDir()：关闭资源

```
26      //关闭资源
27      closedir($r);
```

其他目录操作

1）dirName(一个路径)：得到的是路径的上一层路径

```
//其他函数
$dir1 = 'E:/server/apache/htdocs';
$dir2 = 'E:/server/apache/24file_dir.php';

var_dump(dirname($dir1),dirname($dir2));
```

```
Lighthouse.jpg
Penguins.jpg
string(16) "E:/server/apache" string(16) "E:/server/apache"
```

2）realPath(一个路径)：得到真实路径（目录路径），如果是文件那么得到的结果是 false

```
//其他函数
$dir1 = 'E:/server/apache/htdocs';
$dir2 = 'E:/server/apache/24file_dir.php';

//var_dump(dirname($dir1),dirname($dir2));
var_dump(realpath($dir1),realpath($dir2));
```

```
Lighthouse.jpg
Penguins.jpg
string(23) "E:\server\apache\htdocs" bool(false)
```

3）is_dir()：判断指定路径是否是一个目录

```
//其他函数
$dir1 = 'E:/server/apache/htdocs';
$dir2 = 'E:/server/apache/24file_dir.php';

//var_dump(dirname($dir1),dirname($dir2));
//var_dump(realpath($dir1),realpath($dir2));
var_dump(is_dir($dir1),is_dir($dir2));
```

```
.
..
Chrysanthemum.jpg
image20172017201720170808141
Koala.jpg
Lighthouse.jpg
Penguins.jpg
bool(true) bool(false)
```

4）scandir()：封装版的 opendir\readdir\closedir，获取一个指定路径下的所有文件信息，以数组形式返回

```
//var_dump(is_dir($dir1),is_dir($dir2));


echo '<pre>';
//遍历文件夹
var_dump(scandir('uploads'));
```

```
array(7) {
  [0]=>
  string(1) "."
  [1]=>
  string(2) ".."
  [2]=>
  string(17) "Chrysanthemum.jpg"
  [3]=>
  string(9) "Koala.jpg"
  [4]=>
  string(14) "Lighthouse.jpg"
  [5]=>
  string(12) "Penguins.jpg"
  [6]=>
  string(37) "image20172017201720170808141 40XEF.jpg"
}
```

目录操作

递归遍历目录

递归遍历目录：指定一个目录的情况下，将其下的所有文件和目录，及其目录内部的所有内容都输出出来。

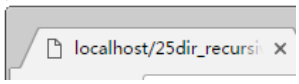
递归算法：将大问题切成相似的小问题（最小单位），然后可以调用解决大问题的方法来解决小问题。
递归函数：函数如果自己内部调用自己，该函数称之为递归函数。

递归遍历目录的思维逻辑
1、 设计一个能够遍历一层文件的函数
a. 创建函数

```
7    $dir = 'uploads';
8
9    /*
10    * 创建函数：能够访问指定路径下的所有文件，且判断出目录还是文件
11    * @param1 string $dir，指定路径
12    */
13   function my_scandir($dir){
14
15   }
```

b. 安全判定：是路径才访问

```
*/
function my_scandir($dir){
    //保证文件安全：如果不是路径没有必要往下
    if(!is_dir($dir))dir($dir . '<br/>');
```



c. 读取全部内容，遍历输出

```
16
17          //读取全部路径信息，遍历输出
18          $files = scandir($dir);
19          foreach($files as $file){
20              //$file就是一个个文件名
21              echo $file . '<br/>';
22          }
23      }
```

2、 找到递归点：遍历得到的文件是目录，应该调用当前函数（调用自己）：
a. 需要构造路径（遍历得到的结果只是文件的名字）

```php
//$file就是一个个文件名
echo $file . '<br/>';

//构造路径
$file_dir = $dir . '/' . $file;
```

b. 需要注意排除.和..

```php
//读取全部路径信息，遍历输出
$files = scandir($dir);
foreach($files as $file){
    //$file就是一个个文件名
    echo $file . '<br/>';

    //排除.和..
    if($file == '.' || $file == '..') continue;

    //构造路径
    $file_dir = $dir . '/' . $file;
    //echo $file_dir . '<br/>';
```

c. 判断是路径还是文件

```php
//构造路径
$file_dir = $dir . '/' . $file;
//echo $file_dir . '<br/>';

//判断路径
if(is_dir($file_dir)){
    //递归点
}
```

d. 递归调用函数

```php
//构造路径
$file_dir = $dir . '/' . $file;
//echo $file_dir . '<br/>';

//判断路径
if(is_dir($file_dir)){
    //递归点
    my_scandir($file_dir);
}
```

3、 找到递归出口：遍历完这个文件夹之后，发现没有任何子文件夹（函数不再调用自己）：自带递归出口

4、 如何显示层级关系？函数第一次运行遍历的结果是最外层目录，内部调用一次说明进入一个子目录，子目录再调用一次函数进行孙子目录…如果能够在第一次调用的时候给个标记，

然后在进入的时候，通过标记的变化来识别层级关系，就可以达到目的：该标记还能代表层次关系：缩进

a. 在函数参数中增加一个标记：默认值为 0

```php
/*
 * 创建函数：能够访问指定路径下的所有文件，且判断出目录还是文件
 * @param1 string $dir，指定路径
 * @param2 int $level = 0，层级，默认顶层
*/
function my_scandir($dir,$level = 0){
    //保证文件安全：如果不是路径没有必要往下
    if(!is_dir($dir))dir($dir . '<br/>');
```

b. 递归调用函数的时候也需要使用该参数：但是是属于当前层级的子层，所以+1

```php
    //判断路径
    if(is_dir($file_dir)){
        //递归点
        my_scandir($file_dir,$level+1);
    }
}
```

c. 根据层级来实现缩进：str_repeat()

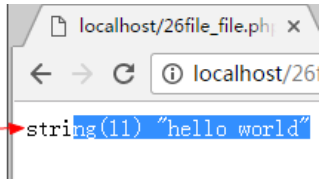
```php
    //读取全部路径信息，遍历输出
    $files = scandir($dir);
    foreach($files as $file){
        //$file就是一个个文件名
        echo str_repeat("  ",$level),$file . '<br/>';

        //排除 和
```

文件操作

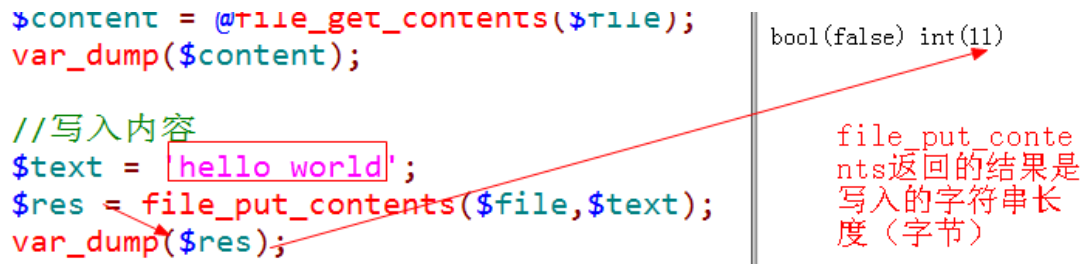
PHP5 常见文件操作函数

1）file_get_contents(文件路径)：获取指定文件的所有内容，如果路径不存在最好做安全处理

```php
//读取内容
$file = 'content.txt';
$content = @file_get_contents($file);
var_dump($content);
```



2）file_put_contents(文件路径，内容)：将指定内容写入到指定文件内：如果当前路径下不存在指定的文件，函数会自动创建（如果路径不存在，不会创建路径）



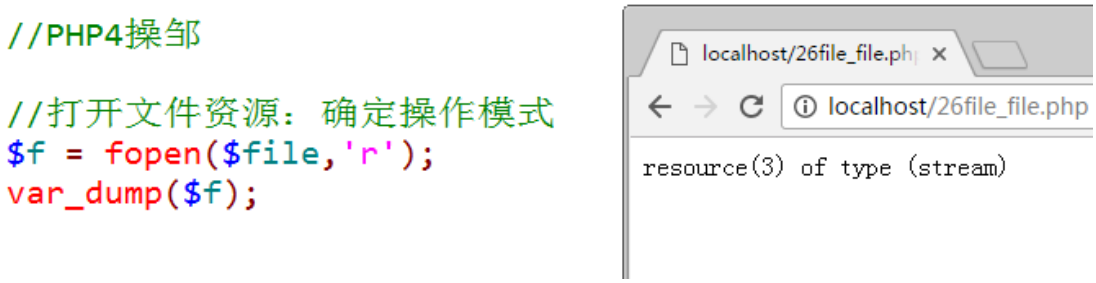
PHP4 常见文件操作函数

PHP4 中是将文件操作用资源形式处理：不论是读还是写都依赖资源指针：文件内容中指针所在位置。

1）fopen(文件路径，打开模式)：打开一个文件资源，限定打开模式

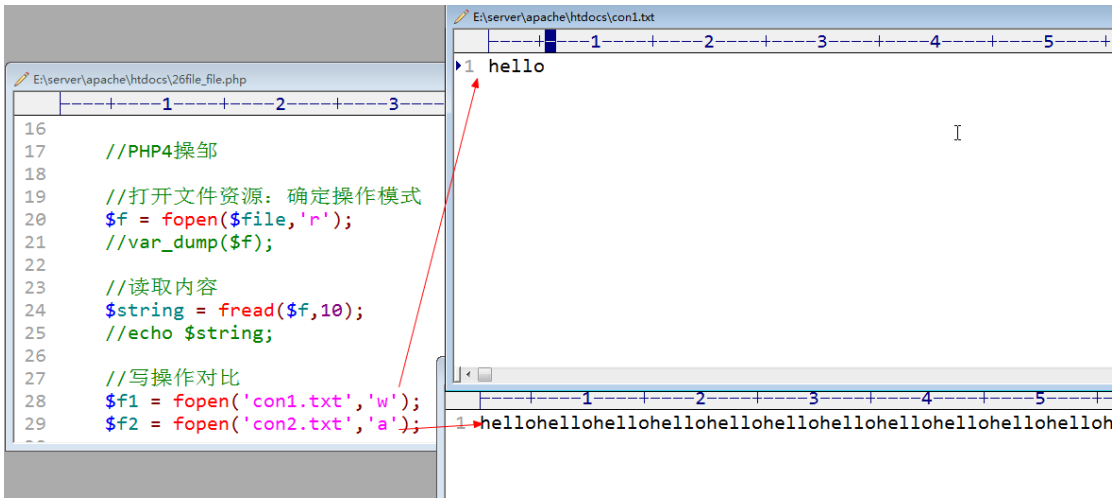
fopen() 中 mode 的可能值列表

mode	说明
'r'	只读方式打开，将文件指针指向文件头。
'r+'	读写方式打开，将文件指针指向文件头。
'w'	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'w+'	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建之。
'a'	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'a+'	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建之。
'x'	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 **fopen()** 调用失败并返回 FALSE，并生成一条 E_WARNING 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 ***open(2)*** 系统调用指定 ***O_EXCL|O_CREAT*** 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。
'x+'	创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 **fopen()** 调用失败并返回 FALSE，并生成一条 E_WARNING 级别的错误信息。如果文件不存在则尝试创建之。这和给 底层的 ***open(2)*** 系统调用指定 ***O_EXCL|O_CREAT*** 标记是等价的。此选项被 PHP 4.3.2 以及以后的版本所支持，仅能用于本地文件。



2）fread(资源，长度)：从打开的资源中读取指定长度的内容（字节）

3）fwrite(资源，内容)：向打开的资源中写入指定的内容



4）fclose(资源)：关闭资源

```
//关闭资源
fclose($f);
```

其他文件操作函数

1）is_file()：判断文件是否正确（不识别路径）
2）filesize()：获取文件大小
3）file_exists()：判断文件是否存在（识别路径）
4）unLink()：取消文件名字与磁盘地址的连接（删除文件）
5）filemtime()：获取文件最后一次修改的时间
6）fseek()：设定 fopen 打开的文件的指针位置
7）fgetc()：一次获取一个字符
8）fgets()：一次获取一个字符串（默认行）
9）file()：读取整个文件，类似 file_get_contents，区别是按行读取，返回一个数组

文件操作

文件下载

文件下载:从服务器将文件通过 HTTP 协议传输到浏览器,浏览器不解析保存成相应的文件。
提供下载方式可以使用 HTML 中的 a 标签：点击下载
1、 缺点 1：a 标签能够让浏览器自动下载的内容有限：浏览器是发现如果解析不了才会启用下载
2、 缺点 2：a 标签下载的文件存储路径会需要通过 href 属性写出来，这样会暴露服务器存储数据的位置（不安全）

PHP 下载：读取文件内容，以文件流的形式传递给浏览器：在响应头中告知浏览器不要解析，激活下载框实现下载。

1） 指定浏览器解析字符集

```
//PHP文件下载

//设定解析字符集
header('Content-type:text/html;charset=utf-8');
```

2） 设定响应头

a） 设定文件返回类型：image/jpg||application/octem-stream

b） 设定返回文件计算方式：Accept-ranges：bytes

c） 设定下载提示：Content-disposition:attachment;filename=’文件名字’

d） 设定文件大小：Accept-length：文件大小（字节）

```
$file = '27file_download.php';

//设定下载响应头
header('Content-type:application/octem-stream');            //以文件流形式传输数据给浏览器
header('Accept-ranges:bytes');                              //以字节方式计算
header('Content-disposition:attachment;filename=' . $file); //附件下载，指定命名
header('Accept-length:' . filesize($file));
```

3） 读取文件

4） 输出文件

方案 1：如果文件较小，可以使用 PHP5 的文件函数操作：file_get_contents

```
//输出文件
//php5：小文件
echo file_get_contents($file);
```

方案 2：文件比较大（网络不好），可以使用 PHP4 的文件操作方式：一次读一点

```php
//php5：小文件
//echo file_get_contents($file);

//php4：大文件
$f = @fopen($file,'r') or die();

//方案1：直接读，然后输出
while($row = fread($f,1024)){
    echo $row;
}

//方案2：判定是否可读，然后再读
/*
while(!feof($f)){
    echo fread($f,1024);
}*/

//关闭资源
fclose($f);
```