php

文件操作，表单处理，正则表达式

一、文件操作

打开关闭文件

1.fopen()
resource **fopen** (string filename, string mode [, bool use_include_path [, resource zcontext]])

fopen()函数将resource绑定到一个流或句柄。绑定之后，脚本就可以通过句柄与此资源交互;

例1:以只读方式打开一个位于本地服务器的文本文件
```
$fh = fopen("test.txt", "r");
```

例2：以只读方式打开一个远程文件
```
$fh = fopen("http://www.baidu.com", "r");
```

文件打开模式

模式	可读?	可写?	文件指针	截断?	创建?
r	是	否	开始	否	否
r+	是	是	开始	否	否
w	否	是	开始	是	是
w+	是	是	开始	是	是
a	否	是	结尾	否	是
a+	是	是	结尾	否	是

文件指针: 指向文件的开头或者是末尾

截断:如果文件已经存在，将文件指针指向文件头并将文件大小截为0.

2.fclose()
bool **fclose** (resource handle)
将 *handle* 指向的文件关闭 。如果成功则返回 **TRUE**，失败则返回 **FALSE**;

文件指针必须有效，并且是通过 fopen() 或 fsockopen() 成功打开的;

虽然每个请求最后都会自动关闭文件，但明确的关闭打开的所有文件是一个好的习惯;

例：$fh = fopen("test.txt", "r");
fclose($fh);

读取文件

php 提供了很多从文件中读取数据的方法，不仅可以一次只读取一个字符，还可以一次读取整个文件。

1.fread()
string **fread** (int handle, int length)

fread()函数从handle指定的资源中读取length个字符,当到达EOF或读取到length个字符时读取将停止。
如果要读取整个文件，使用filesize()函数确定应该读取的字符数;

例：
```
$file = "test.txt";
$fh = fopen( $file, "r");
$str = fread($fh, filesize($file));
echo $str;
fclose($fh);
```

2.fgets()
string **fgets** (int handle [, int length])

fgets()函数从handle指定的资源中读取一行字符。碰到换行符（包括在返回值中）、EOF 或者已经读取了 length - 1 字节后停止（看先碰到那一种情况）；

例：逐行读取文件

```
$handle = fopen("data.txt", "r");
while(!feof($handle)){
    $content = fgets($handle);
    echo $content."<br />”;
}
fclose($handle);
```

注意：length 参数从 PHP 4.2.0 起成为可选项,如果没有指定 length，则默认为 1K，或者说 1024 字节。
从 PHP 4.3 开始，忽略掉 length将继续从流中读取数据直到行结束。

3.file()
array file (string $filename [, int $flags = 0 [, resource $context]])

file()函数将文件读取到数组中，各元素由换行符分隔。

例：
```
$arr = file("test.txt");
print_r($arr);
```

4.file_get_contents()
string **file_get_contents** (string filename [, bool use_include_path [, resource context [, int offset [, int maxlen]]]])

file_get_contents()函数将文件内容读到字符串中;

例：
```
$str = file_get_contents("test.txt");
echo $str;
```

写入文件

1.fwrite()
int fwrite (resource handle, string string [, int length])
fwrite()函数将string的内容写入到由handle指定的资源中。如果指定length参数，将在写入Length个字符时停止。

例：
```
$str = "test text";
$fh = fopen("test.txt", "a");
fwrite($fh, $str);
fclose($fh);
```

2.file_put_contents()
int file_put_contents (string filename, string data [, int flags [, resource context]])
file_put_contents()函数将一个字符串写入文件，与依次调用fopen(),fwrite(),fclose()功能一样;

例：
```
$str = "hello";
file_put_contents("test.txt", $str);
```

复制，重命名，删除文件

1.copy()
bool **copy** (string source, string dest)
将文件从 *source* 拷贝到 *dest*。如果成功则返回 **TRUE**，失败则返回 **FALSE**。
例：Copy("test.txt", "test.txt.bak");

2.rename()
bool **rename** (string oldname, string newname [, resource context])
尝试把 *oldname* 重命名为 *newname*。 如果成功则返回 **TRUE**，失败则返回 **FALSE**。
例：rename("test.txt", “test2.txt”);

3.unlink()
bool **unlink** (string filename)
删除文件，如果删除成功返回true, 否则返回false;
例1：删除一个文本文件
unlink(“test.txt");

读取目录

1.opendir()
resource **opendir** (string path [, resource context])
打开目录句柄;

2.closedir()
void **closedir** (resource dir_handle)
关闭目录句柄

3.readdir()
string **readdir** (resource dir_handle)
返回由dir_handle指定目录中的各个元素。可以使用此函数列出给定目录中的所有文件和子目录

例：
```
$dh = opendir(".");
While($file = readdir($dh)){
   echo $file."<br />";
}
closedir($dh);
```

4.scandir()
array **scandir** (string directory [, int sorting_order [, resource context]])
返回一个包含有 *directory* 中的文件和目录的数组;

5.rmdir()
bool rmdir (string dirname)
删除目录

6.mkdir()
bool mkdir (string pathname [, int mode [, bool recursive [, resource context]]])
尝试新建一个由 pathname 指定的目录。

其他文件操作函数

1.filesize()
int **filesize** (string filename)
取得文件的大小，以字节为单位

2.filectime()
int **filectime** (string filename)
取得文件的创建时间，以unix时间戳方式返回
例：$t = filectime("test.txt");
echo date("Y-m-d H:i:s", $t);

3. fileatime() 返回文件的最后改变时间;

4. filemtime() 返回文件的最后修改时间;

注：“最后改变时间"不同于 "最后修改时间"。最后改变时间指的是对文件inode数据的任何改变，包括改变权限，所属组，拥有者等; 而最后修改时间指的是对文件内容的修改

5. file_exists() 检查文件或目录是否存在，如果存在返回true, 否则返回false;

6. is_readable() 判断文件是否可读，如果文件存在并且可读，则返回true;

7. is_writable() 判断文件是否可写，如果文件存在并且可写，则返回true;

解析目录路径函数

1.basename()
string **basename** (string path [, string suffix])
返回路径中的文件名部份，当指定了可选参数suffix会将这部分内容去掉;
例：$path = "/home/www/data/users.txt";
$filename = basename($path);
$filename2 = basename($path, “.txt");

2.dirname()
string **dirname** (string path)
返回路径中的目录部份;

3.pathinfo()
array **pathinfo** (string path [, int options])
返回一个关联数组，其中包括路径中的四个部份：目录名 dirname，文件名 filename， extension 扩展名，basename文件名+扩展名(如果有扩展名的话)
例：$pathinfo = pathinfo($_SERVER["SCRIPT_FILENAME"]);
print_r($pathinfo);

4.chmod()
修改文件目录权限
chmod() 函数改变文件模式。
chmod — Changes file mode 如果成功则返回 TRUE，否则返回 FALSE。

例：chmod("/somedir/somefile", 755); // 十进制数，可能不对

chmod("/somedir/somefile", "u+rwx,go+rx"); // 字符串，不对

chmod("/somedir/somefile", 0755); // 八进制数，正确的 mode 值

mode 参数包含三个八进制数按顺序分别指定了所有者、所有者所在的组以及所有人的访问限制。每一部分都可以通过加入所需的权限来计算出所要的权限。数字 1 表示使文件可执行，数字 2 表示使文件可写，数字 4 表示使文件可读。

一、表单数据处理

表单简介

GET所有表单输入的数据被加载到请求的URL地址后面;
如：test.php?username=free&password=123&content=dfdsfsfd;
GET方式提交数据只能传递文本，能够提交的数据量大小有限，安全性差;

POST提交数据的方式把表单的数据打包放入http请求中;
POST能够提交更多的数据;

表单提交的数据会自动封装为数组;
用$_GET, $_POST, 或$_REQUEST获得表单提交的数据;

多值表单控件（如复选框和多选框），大大提高了基于web的数据收集能力;
因为这些组件是多值的，所以表单处理函数必须能够识别一个表单变量中可能有多个值;为了让php识别一个表单变量的多个值（即考虑为数组），需要对表单名(元素的name属性值）增加一对中括号，如:

```
<input type="checkbox"name="love[]"/>
```

例：接收用户输入的数据，并保存到相应的文件

文件上传的相关配置

1.表单设置

要进行文件的上传，需要对form表单进行特殊设置;

1.设定表单数据的提交方式为POST

2.设定enctype属性值为: multipart/form-data

3.为了避免用户等待许久之后才发现上传文件太大，可以在表单中添加MAX_FILE_SIZE隐藏域,通过设置其value值可以限制上传文件的大小;

2.PHP设置

1.file_uploads

是否允许通过HTTP上传文件，默认为ON

2.upload_max_filesize

允许上传文件大小的最大值，默认为2M，此指令必须小于post_max_size;

3.upload_tmp_dir

指定上传文件的临时存放路径，这个目录对于拥有此服务器进程的用户必须是可

写的;如果未指定则使用系统默认值;

4.post_max_size

控制POST方式提交数据php所能够接收的最大数据量;

5.memory_limit

指定单个脚本程序可以使用的最大内存容量;

6.max_execution_time

此指令确定php脚本可以执行的最长时间，以秒为单位，默认为30秒;

$_FILES数组

$_FILES超级全局变量作用是存储各种与上传文件有关的信息;

$_FILES是一个二维数组，数组中共有5项：

$_FILES["userfile"]["name"]　　上传文件的名称

$_FILES["userfile"]["type"]　　上传文件的类型

$_FILES["userfile"]["size"]　　上传文件的大小, 以字节为单位

$_FILES["userfile"]["tmp_name"]　　文件上传后在服务器端储存的临时文件名

$_FILES["userfile"]["error"]　　文件上传相关的错误代码

注:userfile只是一个占位符，代表文件上传表单元素的名字; 因此这个值将根据你所给定的名称有所不同;

上传错误信息

$_FILES[‘userfile’][‘error’] 提供了在文件上传过程中出现的错误：

1.UPLOAD_ERR_OK (value = 0)　　如果文件上传成功返回0;

2.UPLOAD_ERR_INI_SIZE (value = 1)
如果试图上传的文件大小超出了 upload_max_filesize指令指定的值，则返回1;

3.UPLOAD_ERR_FORM_SIZE (value = 2)
如果试图上传的文件大小超出了MAX_FILE_SIZE指令（可能嵌入在HTML表单中）指定的值，则返回2;

4.UPLOAD_ERR_PARTIAL (value = 3)
如果文件没有完全上传，则返回3; 如网络出现错误，导致上传过程中断;

5.UPLOAD_ERR_NO_FILE (value = 4)
如果用户没有指定上传的文件就提交表单，则返回4

文件上传函数

1.is_uploaded_file()
bool **is_uploaded_file** (string filename)
is_uploaded_file()函数确定参数filename指定的文件是否使用HTTP POST上传;

```
例：if(is_uploaded_file($_FILES[‘userfile’][‘tmp_name’])){
        copy($_FILES[‘userfile’][‘tmp_name’], “test.txt”);
    }else{
        echo "文件上传失败！";
    }
```

2.move_uploaded_file()
bool **move_uploaded_file** (string filename, string destination)
move_uploaded_file()作用是将上传文件从临时目录移动到目标目录; 虽然copy()也可以实现同样功能，但move_uploaded_file()还提供了一种额外的功能，它将检查由filename输入参数指定的文件确实是通过http post 上传机制上传的。如果所指定的文件并非上传文件，则移动失败，返回false;

```
例：move_uploaded_file($_FILES['userfile']['tmp_name'], "1/test.jpg");
```

三、正则表达式

正则的概念

正则表达式就是一套专门用于处理文本的强大工具；

可以对进行文本查找，匹配，替换;

正则表达式常用于验证表单提交的内容，比如验证电话号码，Email地址，身份证号码等是否有效;

掌握了正则表达式的基础，就可以把知识用到其它的语言(比如:perl,javascript)或者支持Posix风格的正则表达式的UNIX shell环境中;

php支持的正则

Perl风格

PCRE全称为Perl Compatible Regular Expression，意思是Perl兼容正则表达式。PCRE来源于Perl语言，而Perl是对字符串操作功能最强大的语言之一，PHP的最初版本就是由Perl开发的产品。

在PCRE中，通常将正则表达式 **包含在两个反斜线"/"之间;**

例："/apple/"

定界符也不仅仅局限于"/"。除了字母、数字和斜线"\"以外的任何字符都可以作为定界符，像"#"、"/"、"!"等都可以的。

Posix 风格

一般而言，实现相同的功能Perl风格的，效率高些，我们一般使用Perl风格的函数！

例:检查email地址的合法性

1.用字符串查找的方法:

```
function is_email($email)
{
$has_at = strpos($email, "@");    //检查是否 包含@
$has_dot = strpos($email, ".");   //检查是否包含.

if($has_at && $has_dot ){
        return true;
} else{
        return false
}
}
echo is_email("tom@php.net");       //true
echo is_email("tom@php");           //false
```

Perl 兼容正则函数

1. preg_match() 进行正则表达式匹配

2. preg_match_all() 进行正则表达式全局匹配

3. preg_replace() 执行正则表达式的搜索和替换

4.preg_split() 用正则表达式分割字符串

5.preg_grep() 返回与模式匹配的数组单元

元字符

*	匹配前面的字符零次或多次 等同于 {0，}
+	匹配前面的字符一次或多次 等同于 {1，}
?	匹配前面的字符零次或一次 等同于 {0，1}
|	匹配两个或多个选择
^	匹配字符串的开始位置
$	匹配字符串结束位置
\b	匹配单词的边界(如空格、横杠，但不包括下划线)
\B	匹配除单词边界以外的部分
[]	匹配方括号中的任一字符
[^]	匹配除方括号中的字符外的任何字符
{m}	m 是一个非负整数。匹配确定的 m 次
{m,}	m 是一个非负整数。至少匹配m 次
{m,n}	最少匹配 m次且最多匹配 n次
()	表示一个整体
.	匹配除换行之外的任何一个字符

与定义字符

由于某些模式会反复用到，所以可以使用以下预定义类;

\d	匹配一个数字；等价于[0-9]
\D	匹配除数字以外任何一个字符；等价于[^0-9]
\w	匹配一个英文字母、数字或下划线；等价于[0-9a-zA-Z_]
\W	匹配除英文字母、数字和下划线以外任何一个字符；等价于[^0-9a-zA-Z_]
\s	匹配一个空白字符；等价于[\f\n\r\t\v]
\S	匹配除空白字符以外任何一个字符；等价于[^\f\n\r\t\v]

匹配开始与结束

在某些情况下，需要对匹配范围进行限定，以获得更准确的匹配结果；

"^"置于字符串的开始确保模式匹配出现在字符串首端；

"$"置于字符串的结束，确保模式匹配出现字符串尾端。

如果不加边界限制元字符，将获得更多的匹配结果。

例：
```
$res = preg_match("/^hello/", "hello world");
var_dump($res);
$res = preg_match("/world$/", "hello world");
var_dump($res);
```

匹配任意字符

用"."匹配除换行符外任何一个字符

例：
```
$res = preg_match('/./', "something");
var_dump($res);
```

通常可以使用".*"组合来匹配除换行符外的任何字符

匹配范围内字符

用"[start-end]"匹配包含某个范围的字符

[a-z]匹配所有的小写字母
[A-Z]　　匹配所有的大写字母
[a-zA-Z]　匹配所有的字母
[0-9]　　匹配所有的数字
[0-9\.\-]　匹配所有的数字，句号和减号

例：$res = preg_match("^[a-z]+$", "abc");
　　var_dump($res);

重复匹配

正则表达式中有一些用于重复匹配的元字符："?"、"*"、"+"。他们主要的区别是重复匹配的次数不同。

"?"　匹配前面的字符零次或一次　　等同于　{0，1}

"*"　匹配前面的字符零次或多次　　等同于　{0，}

"+"　匹配前面的字符一次或多次　　等同于　{1，}

"{m}"　匹配确定的 m 次。 m 是一个非负整数;

"{m，n}" 最少匹配 m 次且最多匹配 n 次。 m 和 n 均为非负整数，其中m <= n;

"{m，}"　　至少匹配m 次。 m 是一个非负整数;

匹配多个选择

用圆括号"(word1|word2|...)"定义包含word1、word2、...的任意字符串的规则

例：
```
$res= preg_match ('/^(this|the)/', 'this island is a beautiful land');
var_dump($res);
```

模式修正符

i	在和模式进行匹配时不区分大小写
m	将字符串视为多行
s	模式中的圆点元字符 “ .“将匹配所有的字符，包括换行符
x	模式中的空白忽略不计，除非已经转义
U	取消贪婪匹配

模式修正符在正则表达式定界符之外使用。

例：
```
$str='21PhP8';
 //不区分大小写
 $preg='/php/i';
//拿到所有匹配的
$res= preg_match($preg,$str);
var_dump($res);
```

谢　谢