Php 学习

第一种特殊类型—资源

资源（resource）：资源是由专门的函数来建立和使用的，例如打开文件、数据连 接、图形画布。我们可以对资源进行操作（创建、使用和释放）。任何资源，在不需要的时候应该被及时释放。如果我们忘记了释放资源，系统自动启用垃圾回收机 制，在页面执行完毕后回收资源，以避免内存被消耗殆尽。

举例如下：

```
<?php
$file=fopen("f.txt","r");   //打开文件
$con=mysql_connect("localhost","root","root");  //连接数据库
$img=imagecreate(100,100);//图形画布
?>
```

1、php 字符串合并函数 implode()

函数说明：implode(分隔符[可选], 数组)

返回值：把数组元素组合为一个字符串

例子：

```
$arr = array('Hello', 'World!');
$result = implode('', $arr);
print_r($result);//结果显示 Hello World!
```

2、php 字符串分隔函数 explode()

函数说明：explode(分隔符[可选], 字符串)

返回值：函数返回由字符串组成的数组

例子：

```
$str = 'apple,banana';
$result = explode(',', $str);
print_r($result);//结果显示 array('apple','banana')
```

字符串的转义

php 字符串转义函数 addslashes()

函数说明：用于对特殊字符加上转义字符，返回一个字符串

返回值：一个经过转义后的字符串

例子：

```
$str = "what's your name?";
echo addslashes($str);//输出：what\'s your name?
```

正则表达式的基本语法

PCRE 库函数中，正则匹配模式使用分隔符与元字符组成，分隔符可以是非数字、非反斜线、非空格的任意字符。经常使用的分隔符是正斜线(/)、hash 符号(#) 以及取反符号(~)，例如：

```
/foo bar/
#^[^0-9]$#
~php~
```

如果模式中包含分隔符，则分隔符需要使用反斜杠（\）进行转义。

```
/http:\/\//
```

如果模式中包含较多的分割字符，建议更换其他的字符作为分隔符，也可以采用 preg_quote 进行转义。

```
$p = 'http://';
$p = '/'.preg_quote($p, '/').'/';
echo $p;
```

分隔符后面可以使用模式修饰符，模式修饰符包括：i, m, s, x 等，例如使用 i 修饰符可以忽略大小写匹配：

```
$str = "Http://www.imooc.com/";
if (preg_match('/http/i', $str)) {
  echo '匹配成功';
}
```

```
<?php
$subject = "my email is spark@imooc.com";
//在这里补充代码，实现正则匹配，并输出邮箱地址
$pat = '/\w+@(\w+\.)+\w+/';
preg_match($pat, $subject, $match);
echo $match[0];
```

Cookie 是存储在客户端浏览器中的数据，我们通过 Cookie 来跟踪与存储用户数据。一般情况 下，Cookie 通过 HTTP headers 从服务端返回到客户端。多数 web 程序都支持 Cookie 的操作，因为 Cookie 是存在于 HTTP 的标头之中，所以必须在其他信息输出以 前进行设置，类似于 header 函数的使用限制。

PHP 通过 setcookie 函数进行 Cookie 的设置，任何从浏览器发回的 Cookie，PHP 都会自动的将他存储在$_COOKIE 的全局变量之中，因此我们可以通过$_COOKIE['key']的形式来读取某个 Cookie 值。

PHP 中的 Cookie 具有非常广泛的使用，经常用来存储用户的登录信息，购物车等，且在使用会话 Session 时通常使用 Cookie 来存储会话 id 来识别用户，Cookie 具备有效期，当有效期结束之后，Cookie 会自动的从客户端删除。同时为了进行安全控制，Cookie 还可以设置域跟路 径，我们会在稍后的章节中详细的讲解他们。

PHP 设置 Cookie 最常用的方法就是使用 setcookie 函数，setcookie 具有 7 个可选参数，我们常用到的为前 5 个：

name（ Cookie 名）可以通过$_COOKIE['name'] 进行访问
value（Cookie 的值）
expire（过期时间）Unix 时间戳格式，默认为 0，表示浏览器关闭即失效
path（有效路径）如果路径设置为'/'，则整个网站都有效
domain（有效域）默认整个域名都有效，如果设置了'www.imooc.com',则只在 www 子域中有效

```
$value = 'test';
setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);  //有效期一小时
setcookie("TestCookie", $value, time()+3600, "/path/", "imooc.com"); //设置路径与域
```

PHP 中还有一个设置 Cookie 的函数 setrawcookie，setrawcookie 跟 setcookie 基本一样，唯一的不同就是 value 值不会自动的进行 urlencode，因此在需要的时候要手动的进行 urlencode。

```
setrawcookie('cookie_name', rawurlencode($value), time()+60*60*24*365);
```

因为 Cookie 是通过 HTTP 标头进行设置的，所以也可以直接使用 header 方法进行设置。

```
<?php
$value = time();
//在这里设置一个名为 test 的 Cookie
setcookie("test",$value);
if (isset($_COOKIE['test'])) {
   echo 'success';
}
```

cookie 的删除与过期时间

通过前面的章节，我们了解了设置 cookie 的函数，但是我们却发现 php 中没有删除 Cookie 的函数，在 PHP 中删除 cookie 也是采用 setcookie 函数来实现。

```
setcookie('test', '', time()-1);
```

可以看到将 cookie 的过期时间设置到当前时间之前，则该 cookie 会自动失效，也就达到了删除 cookie 的目的。之所以这么设计是因为 cookie 是通过 HTTP 的标头来传递的，客户端根据服务端返回的 Set-Cookie 段来进行 cookie 的设置，如果删除 cookie 需要使用新的 Del-Cookie 来实现，则 HTTP 头就会变得复杂，实际上仅通过 Set-Cookie 就可以简单明了的实现 Cookie 的设置、更新与删除。

了解原理以后，我们也可以直接通过 header 来删除 cookie。

```
header("Set-Cookie:test=1393832059; expires=".gmdate('D, d M Y H:i:s \G\M\T', time()-1));
```

这里用到了 gmdate，用来生成格林威治标准时间，以便排除时差的影响。

cookie 的有效路径

cookie 中的路径用来控制设置的 cookie 在哪个路径下有效，默认为'/'，在所有路径下都有，当设定了其他路径之后，则只在设定的路径以及子路径下有效，例如：

```
setcookie('test', time(), 0, '/path');
```

上面的设置会使 test 在/path 以及子路径/path/abc 下都有效，但是在根目录下就读取不到 test 的 cookie 值。

一般情况下，大多是使用所有路径的，只有在极少数有特殊需求的时候，会设置路径，这种情况下只在指定的路径中才会传递 cookie 值，可以节省数据的传输，增强安全性以及提高性能。

当我们设置了有效路径的时候，不在当前路径的时候则看不到当前 cookie。

```
setcookie('test', '1',0, '/path');
var_dump($_COOKIE['test']);
```

session 与 cookie 的异同

cookie 将数据存储在客户端，建立起用户与服务器之间的联系，通常可以解决很多问题，但是 cookie 仍然具有一些局限：

cookie 相对不是太安全，容易被盗用导致 cookie 欺骗
单个 cookie 的值最大只能存储 4k
每次请求都要进行网络传输，占用带宽

session 是将用户的会话数据存储在服务端，没有大小限制，通过一个 session_id 进行用户识别，PHP 默认情况下 session id 是通过 cookie 来保存的，因此从某种程度上来说，seesion 依赖于 cookie。但这不是绝对的，session id 也可以通过参数来实现，只要能将 session id 传递到服务端进行识别的机制都可以使用 session。

使用 session

在 PHP 中使用 session 非常简单，先执行 session_start 方法开启 session，然后通过全局变量 $_SESSION 进行 session 的读写。

```
session_start();
$_SESSION['test'] = time();
var_dump($_SESSION);
```

session 会自动的对要设置的值进行 encode 与 decode，因此 session 可以支持任意数据类型，包括数据与对象等。

```
session_start();
$_SESSION['ary'] = array('name' => 'jobs');
```

```
$_SESSION['obj'] = new stdClass();
var_dump($_SESSION);
```

默认情况下，session 是以文件形式存储在服务器上的，因此当一个页面开启了 session 之后，会独占这个 session 文件，这样会导致当前用户的其他并发访问无法执行而等待。可以采用缓存或者数据库的形式存储来解决这个问题，这个我们会在一些高级的课程中讲到。

```
<?php
//在这里设置 name 的 session 值为 jobs
session_start();
$_SESSION['name'] = 'jobs';
echo "session_id:".session_id();
echo "</br>";
echo $_SESSION['name'];
```