PHP常用功能块

网址：www.lampbrother.net
电话：400 700 1307

无兄弟 不编程

PHP常用功能块

1. 错误和异常处理
2. 日期和时间
3. 动态图像处理

一、错误和异常处理

- 1.1 错误类型和基本的调试方法
- 1.2 错误日志
- 1.3 异常处理

1.1 错误类型和基本的调试方法

- PHP程序的错误发生一般归属于下列三个领域:
  - **语法错误:**
    - 语法错误最常见，并且也容易修复。如：代码中遗漏一个分号。这类错误会阻止脚本的执行。
  - **运行时错误:**
    - 这种错误一般不会阻止PHP脚本的执行，但会阻止当前要做的事情。输出一条错误，但php脚本继续执行
  - **逻辑错误:**
    - 这种错误最麻烦，既不阻止脚本执行，也不输出错误消息。
- 一个异常则是在一个程序执行过程中出现的一个例外，或是一个事件，它中断了正常指令的运行，跳转到其他程序模块继续执行。

PHP的错误报告级别

- E_ALL //所有信息 值: 6143
- E_ERROR //致命的运行时错误 值: 1
- E_RECOVERABLE_ERROR //接近致命的运行时错误，若未被捕获则视同E_ERROR 值: 4096
- E_WARNING //运行时警告(非致命性错误) 值: 2
- E_PARSE //编译时解析错误 值: 4
- E_NOTICE //运行时提醒(经常是bug，也可能是有意的) 值: 8
- E_STRICT //编码标准化警告(建议如何修改以向前兼容) 值: 2048
- E_CORE_ERROR //PHP启动时初始化过程中的致命错误 值: 16
- E_CORE_WARNING // PHP启动时初始化过程中的警告(非致命性错)值: 32
- E_COMPILE_ERROR // 编译时致命性错 值: 64
- E_COMPILE_WARNING //编译时警告(非致命性错) 值: 128
- E_USER_ERROR //用户自定义的致命错误 值: 256
- E_USER_WARNING // 用户自定义的警告(非致命性错误) 值: 512
- E_USER_NOTICE //用户自定义的提醒(经常是bug) 值: 1024

php.ini配置文件

- display_errors: 是否开启PHP输出错误报告的功能
  - 值为: On（默认输出错误报告）、 Off（屏蔽所有错误信息）
  - 在PHP脚本中可调用ini_set( )函数，动态设置php.ini配置文件.
  - 如: ini_set("display_errors","On"); //显示所有错误信息
- error_reporting: 设置不同的错误报告级别。
  - error_reporting = E_ALL & ~E_NOTICE
    -- 可以抛出任何非注意的错误，默认值
  - error_reporting = E_ERROR | E_PARSE | E_CORE_ERROR
    -- 只考虑致命的运行时错误、新解析错误和核心错误。
  - error_reporting = E_ALL & ~(E_USER_ERROR | E_USER_WARNING | E_USER_NOTICE)
    -- 报告除用户导致的错误之外的所有错误。
  - 在PHP脚本可以通过error_reporting( )函数动态设置错误报告级别。如: error_reporting(E_ALL);

设置错误级别实例：error.php

```
<h2>测试错误报告</h2>
<?php
/*开启php.ini中的display_errors指令，只有该指令开启如有错误报告才能输出*/
  ini_set('display_errors',1);
  /*通过error_reporting()函数设置在本脚本中，输出所有级别的错误报告*/
  error_reporting(E_ALL);
/*“注意(notice)”的报告，不会阻止脚本的执行，并且不一定是一个问题 */
  getType($var);              //调用函数时提供的参数变量没有在之前声明
/*“警告(warning)”的报告，指示一个问题，但是不会阻止脚本的执行 */
  getType();                  //调用函数时没有提供必要的参数
  /*“错误(error)”的报告，它会终止程序，脚本不会再向下执行 */
  get_Type();                 //调用一个没有被定义的函数
?>
```

PHP错误报告行为的配置指令

- display_startup_errors = Off
  - 是否显示PHP引擎在初始化时遇到的错误。
- log_errors = On
  - 决定日志语句记录的位置。
- error_log （默认null）
  - 指定错误写进的文件或记录错误日志于系统日志syslog。
- Log_errors_max_len=1024
  - 每个日志项的最大长度，单位是字节。0表示最大。

1.2 错误日志

- 两种方式记录错误日志:
  - 使用指定的文件记录错误报告日志
  - 错误日志记录到操作系统的日志里

使用指定的文件记录错误报告日志

- 1、先配置php.ini:
  - error_reporting = E_ALL //将向PHP发送每个错误
  - display_errors=Off //不显示错误报告
  - log_errors=On //决定日志语句记录的位置。
  - log_errors_max_log=1024 // 每个日志项的最大长度
  - error_log=G:/myerror.log //指定错误写进的文件
- 2、使用函数: 在php文件中使用error_log()来记录日志,就可以将信息写入到myerror.log文件中
  - 如: error_log("登录失败了！ ");

使用指定的文件记录错误报告日志

```php
<?php
    if(!Ora_Logon($username, $password)){
        error_log("Oracle数据库不可用!", 0);
        //将错误消息写入到操作系统日志中
    }
    if(!($foo=allocate_new_foo()){
        error_log("出现大麻烦了!", 1,
    "webmaster@www.mydomain.com");    //发送到管理员邮箱中
    }
    error_log("搞砸了!",    2,    "localhost:5000");
        //发送到本机对应5000端口的服务器中
    error_log("搞砸了!",    3,    "/usr/local/errors.log");
        //发送到指定的文件中
?>
```

错误日志记录到操作系统的日志里

- 1、先配置PHP.ini文件中
  - error_reporting = E_ALL　　//将向PHP发送每个错误
  - display_errors=Off　　//不显示错误报告
  - log_errors=On　　//决定日志语句记录的位置。
  - log_errors_max_log=1024　　// 每个日志项的最大长度
  - error_log=syslog　　//指定到系统日志中。
- 2、使用四个函数来记录日志:
  - define_syslog_variables(); //为系统日志初始化配置
  - openlog(); //打开一个日志链接
  - syslog(); //发送一条日志记录
  - closelog(); //关闭日志链接

错误日志记录到操作系统的日志里

```
<?php
   define_syslog_variables();
   openlog("PHP5", LOG_PID , LOG_USER);
   syslog(LOG_WARNING, "警告报告向syslog中发送的演示
   , 警告时间: ".date("Y/m/d H:i:s"));
   closelog();
?>
```

查看日志：如windows系统，通过右击“我的电脑”-> 选择管理选项->在系统工具菜单中选择事件查看器->在应用程序选项中即可看到日志了。

1.3 异常处理

- 异常（Exception）处理用于在指定的错误发生时改变脚本的正常流程。是PHP5中的一个新的重要特性。异常处理是一种可扩展、易维护的错误处理统一机制，并提供了一种新的面向对象的错误处理方式。
- 异常处理格式:

```
try{
        使用try去包含可能会发生异常的代码.
        一旦出现异常try进行捕获异常，交给catch处理。
        抛出异常语句: throw 异常对象。
}catch（异常对象参数）{
        在这里做异常处理。
}[catch（。，，）{
        .. .. ..
}]
```

一个简单异常处理实例

```php
<?php
    try {
        $error = 'Always throw this error';
        throw new Exception($error);
                            //创建一个异常对象，通过throw语句抛出
        echo 'Never executed';
                    //从这里开始，try代码块内的代码将不会再被执行
    } catch (Exception $e) {
        echo 'Caught exception: ',  $e->getMessage(), "\n";
                                //输出捕获的异常消息
    }
    echo 'Hello World';          //程序没有崩溃继续向下执行
?>
```

系统自带异常处理

```php
<?php
    class Exception{
        protected $message = 'Unknown exception';   // 异常信息
        protected $code = 0;                        // 用户自定义异常代码
        protected $file;                            // 发生异常的文件名
        protected $line;                            // 发生异常的代码行号

        function __construct($message = null, $code = 0);

        final function getMessage();               // 返回异常信息
        final function getCode();                  // 返回异常代码
        final function getFile();                  // 返回发生异常的文件名
        final function getLine();                  // 返回发生异常的代码行号
        final function getTrace();                 // backtrace() 数组
        final function getTraceAsString();         // 已格成化成字符串
    的 getTrace() 信息

        /* 可重载的方法 */
        function __toString();                     // 可输出的字符串
    }
?>
```

自定义异常

```php
<?php
    /* 自定义的一个异常处理类，但必须是扩展内异常处理类的子类 */
    class MyException extends Exception{
        //重定义构造器使第一个参数 message 变为必须被指定的属性
        public function __construct($message, $code=0){
            //可以在这里定义一些自己的代码
         //建议同时调用 parent::construct()来检查所有的变量是否已被赋值
            parent::__construct($message, $code);
        }
        public function __toString() {
          //重写父类方法，自定义字符串输出的样式
          return __CLASS__.": [".$this->code."]: ".$this->message."<br>";
        }
        public function customFunction() {
            //为这个异常自定义一个处理方法
            echo "按自定义的方法处理出现的这个类型的异常<br>";
        }
    }
?>
```

自定义异常

```
<?php
    try {              //使用自定义的异常类捕获一个异常，并处理异常
        $error = '允许抛出这个错误';
        throw new MyException($error);
         //创建一个自定义的异常类对象，通过throw语句抛出
        echo 'Never executed';                  //从这里开始，try代码块内的
    代码将不会再被执行
    } catch (MyException $e) {                  //捕获自定义的异常对象
        echo '捕获异常: '.$e;                    //输出捕获的异常消息
        $e->customFunction();                   //通过自定义的异常对象中的方
    法处理异常
    }
    echo '你好呀';                               //程序没有崩溃继续向下执行
?>
```

捕获多个异常

- 在try代码之后，必须至少给出一个catch代码块，也可以将多个catch代码块与一个try代码块关联使用。那么使用多个catch就可以捕获不同的类所产生的异常。注意顺序。

二、日期和时间

- 2.1 UNIX时间戳
- 2.2 在PHP中获取日期和时间
- 2.3 日期和时间的格式化输出
- 2.4 修改PHP的默认时区
- 2.5 使用微秒计算PHP脚本执行时间

2.1 UNIX时间戳

- Unix 时间戳：
  - 自从 Unix 纪元（格林威治时间 1970 年 1 月 1 日 00:00:00）到当前时间的秒数 .
- 相关函数:
- 使用time( )函数返回一个当前系统的时间戳
- mktime( ) -- 取得一个日期的 Unix 时间戳
  - 格式: int mktime(时[,分[,秒[,月[,日[,年[,is_dst区]]]]]]);
  - 注意: is_dst参数表示是否为夏时制，在PHP5.10以后此参数已废除。

2.2 在PHP中获取日期和时间

- getdate -- 取得日期／时间信息
  - 格式: array getdate ([int timestamp])
  - 返回一个根据 *timestamp* 得出的包含有日期信息的结合数组。如果没有给出时间戳则认为是当前本地时间。数组中的单元如下:

表格 1. 返回的关联数组中的键名单元

键名	说明	返回值例子
"seconds"	秒的数字表示	*0* 到 *59*
"minutes"	分钟的数字表示	*0* 到 *59*
"hours"	小时的数字表示	*0* 到 *23*
"mday"	月份中第几天的数字表示	*1* 到 *31*
"wday"	星期中第几天的数字表示	*0*（表示星期天）到 *6*（表示星期六）
"mon"	月份的数字表示	*1* 到 *12*
"year"	4 位数字表示的完整年份	例如：*1999* 或 *2003*
"yday"	一年中第几天的数字表示	*0* 到 *365*
"weekday"	星期几的完整文本表示	*Sunday* 到 *Saturday*
"month"	月份的完整文本表示	*January*> 到 *December*
0	自从 Unix 纪元开始至今的秒数，和 time() 的返回值以及用于 date() 的值类似。	系统相关，典型值为从 *-2147483648* 到 *2147483647*。

2.3 日期和时间的格式化输出

- date -- 格式化一个本地时间/日期
- 格式: string date (string format [, int timestamp])
  - 返回将整数 *timestamp* 按照给定的格式字串而产生的字符串。如果没有给出时间戳则使用本地当前时间。换句话说，*timestamp* 是可选的，默认值为 time()（当前时间戳）。
  - 例如:

```
echo date("Y年m月d日 H:i:s"); //2010年10月28日 14:22:28
```

Y: 四位数年　m: 月01-12　n:月1-12　d: 天01-31　j: 天1-31

H: 时24时制　h: 小时12制　i: 分钟00-59　s: 秒00-59　w: 星期几0-6

A: 上午AM或下午PM　a: 上午am或下午pm。

2.4 修改PHP的默认时区

- 修改PHP的默认时区有两种方式:
- 1、修php.ini配置文件:
  - date.timezone = Etc/GMT+8
- 2、使用date_default_timezone_set函数: -- 设定用于一个脚本中所有日期时间函数的默认时区 。
  - 如: date_default_timezone_set(“PRC”);//中国时区。
  - date_default_timezone_get ( ) ; //获取当前时区

2.5 使用微秒计算PHP脚本执行时间

- microtime -- 返回当前 Unix 时间戳和微秒数
  - 格式: mixed microtime ([bool get_as_float])
- microtime() 当前 Unix 时间戳以及微秒数。本函数仅在支持 gettimeofday() 系统调用的操作系统下可用。
- 如果调用时不带可选参数，本函数以 "msec sec" 的格式返回一个字符串，其中 sec 是自 Unix 纪元（0:00:00 January 1, 1970 GMT）起到现在的秒数，msec 是微秒部分。字符串的两部分都是以秒为单位返回的。
- 如果给出了 *get_as_float* 参数并且其值等价于 TRUE，microtime() 将返回一个浮点数。

三、动态图像处理

- 3.1 PHP中GD库的使用
- 3.2 画布管理
- 3.3 设置颜色
- 3.4 生成图片
- 3.5 绘制图像
- 3.6 在图像中绘制文字
- 3.7 在PHP中实现验证码类的设计

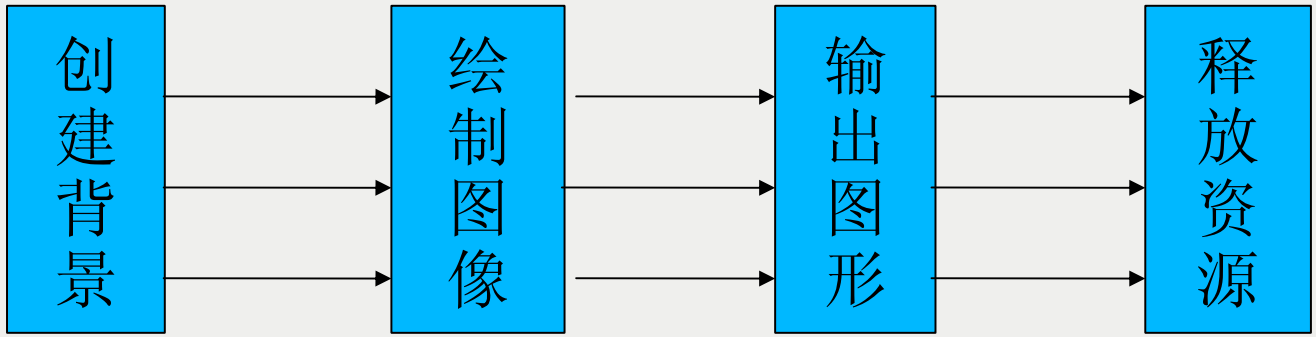
3.1 PHP中GD库的使用

- PHP 不仅限于只产生 HTML 的输出，还可以创建及操作多种不同格式的图像文件。PHP提供了一些内置的图像信息函数，也可以使用GD函数库创建新图像或处理已有的图像。目前GD2库支持JPEG、PNG和WBMP格式。但不再支持GIF格式。
  - JPEG 是一种压缩标准的名字，通常是用来存储照片或者存储具有丰富色彩和色彩层次的图像。这种格式使用了有损压缩。
  - PNG 是可移植的网络图像，对图像采用了无损压缩标准。
  - WBMP是专门为无线通讯设备设计的文件格式。但是并没有得到广泛应用。

图像的生成步骤

在PHP中创建一个图像应该完成如下所示的4个步骤：

1. 创建一个背景图像（也叫画布），以后的操作都基于此背景图像。
2. 在背景上绘制图像轮廓或输入文本。
3. 输出最终图形
4. 释放资源



```php
<?php
  // 创建背景图像
  $height = 200;
  $width = 200;
  $im = ImageCreateTrueColor($width, $height);      //建立空白背景
  $white = ImageColorAllocate ($im, 255, 255, 255);//设置绘图颜色
  $blue = ImageColorAllocate ($im, 0, 0, 64);
  imageFill($im, 0, 0, $blue);                      //绘制背景
  imageLine($im, 0, 0, $width, $height, $white);    //画线
  imageString($im, 4, 50, 150, 'Sales', $white);    //添加字串
  header('Content-type: image/png');
  imagePng ($im);                                   //以 PNG 格式将图像输出
  imageDestroy($im);
?>
```

3.2 画布管理

- imagecreate -- 新建一个基于调色板的图像
  - resource imagecreate (int x_size, int y_size)
  - 本函数用来建立空新画布，参数为图片大小，单位为像素(pixel)。支持256色。
- imagecreatetruecolor -- 新建一个真彩色图像
  - resource imagecreatetruecolor (int x_size, int y_size)
  - 新建一个真彩色图像画布 ，需要 GD 2.0.1 或更高版本，不能用于 GIF 文件格式。
- imagedestroy -- 销毁一图像
  - bool imagedestroy (resource image)
  - imagedestroy() 释放与 *image* 关联的内存。

其他函数

- resource imagecreatefrompng (string filename)
  从 PNG 文件或 URL 新建一图像
- resource imagecreatefromjpeg (string filename)
- 从 JPEG 文件或 URL 新建一图像
- resource imagecreatefromgif (string filename
- 从 GIF 文件或 URL 新建一图像
- resource imagecreatefromwbmp (string filename)
  从 WBMP 文件或 URL 新建一图像
- int imagesx (resource image) --- 取得图像宽度
- nt imagesy (resource image) --- 取得图像高度

3.3 设置颜色

- imagecolorallocate -- 为一幅图像分配颜色
- **语法: int imagecolorallocate (resource image, int red, int green, int blue)**
- imagecolorallocate() 返回一个标识符，代表了由给定的 RGB 成分组成的颜色。*image* 参数是
- imagecreatetruecolor() 函数的返回值。*red*，*green* 和 *blue* 分别是所需要的颜色的红，绿，蓝成分。这些参数是 0 到 255 的整数或者十六进制的 0x00 到 0xFF。imagecolorallocate() 必须被调用以创建每一种用在 *image* 所代表的图像中的颜色。

3.4 生成图片

- imagegif -- 以 GIF 格式将图像输出到浏览器或文件
  - 语法: bool imagegif (resource image [,string filename])
- imagejpeg -- 以 JPEG 格式将图像输出到浏览器或文件
  - 语法: bool imagejpeg (resource image [,string filename [, int quality]])
- imagepng -- 以 PNG 格式将图像输出到浏览器或文件
  - 语法: bool imagepng (resource image [,string filename])
- imagewbmp -- 以 WBMP 格式将图像输出到浏览器或文件
  - 语法: bool imagewbmp (resource image [, string filename [, int foreground]])

3.5 绘制图像

- imagefill -- 区域填充
  - 语法: bool imagefill(resource image,int x,int y, int color)
  - imagefill() 在 *image* 图像的坐标 *x*, *y*（图像左上角为 0, 0）处用 *color* 颜色执行区域填充（即与 x, y 点颜色相同且相邻的点都会被填充）。
- imagesetpixel -- 画一个单一像素
  - 语法: bool imagesetpixel (resource image, int x, int y, int color)
  - imagesetpixel() 在 *image* 图像中用 *color* 颜色在 *x*, *y* 坐标（图像左上角为 0, 0）上画一个点。
- imageline -- 画一条线段
  - 语法: bool imageline (resource image, int x1, int y1, int x2, int y2, int color)
  - imageline() 用 *color* 颜色在图像 *image* 中从坐标 *x1*, *y1* 到 *x2*, *y2*（图像左上角为 0, 0）画一条线段。

- imagerectangle -- 画一个矩形
  - 语法: bool imagerectangle (resource image, int x1, int y1, int x2, int y2, int col)
  - imagerectangle() 用 *col* 颜色在 *image* 图像中画一个矩形，其左上角坐标为 x1，y1，右下角坐标为 x2，y2。图像的左上角坐标为 0，0。
- imagefilledrectangle -- 画一矩形并填充
  - 语法: bool imagefilledrectangle (resource image, int x1, int y1, int x2, int y2, int color)
  - imagefilledrectangle() 在 *image* 图像中画一个用 *color* 颜色填充了的矩形，其左上角坐标为 *x1*，*y1*，右下角坐标为 *x2*，*y2*。0，0 是图像的最左上角。

- imagepolygon -- 画一个多边形
  - 语法: bool imagepolygon (resource image, array points, int num_points, int color)
  - imagepolygon() 在图像中创建一个多边形。*points* 是一个 PHP 数组，包含了多边形的各个顶点坐标，即 points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, 以此类推。*num_points* 是顶点的总数。
- imagefilledpolygon -- 画一多边形并填充
  - 语法: bool imagefilledpolygon (resource image, array points, int num_points, int color)
  - imagefilledpolygon() 在 *image* 图像中画一个填充了的多边形。*points* 参数是一个按顺序包含有多边形各顶点的 *x* 和 *y* 坐标的数组。 *num_points* 参数是顶点的总数，必须大于 3。

- imageellipse -- 画一个椭圆
  - 语法: bool imageellipse (resource image, int cx, int cy, int w, int h, int color)
  - imageellipse() 在 *image* 所代表的图像中画一个中心为 *cx*, *cy*（图像左上角为 0, 0）的椭圆。*w* 和 *h* 分别指定了椭圆的宽度和高度，椭圆的颜色由 *color* 指定。
- imagefilledellipse -- 画一椭圆并填充
  - 语法: bool imagefilledellipse (resource image, int cx, int cy, int w, int h, int color)
  - imagefilledellipse() 在 *image* 所代表的图像中以 *cx*, *cy*（图像左上角为 0, 0）为中心画一个椭圆。*w* 和 *h* 分别指定了椭圆的宽和高。椭圆用 *color* 颜色填充。如果成功则返回 TRUE，失败则返回 FALSE。

- imagearc -- 画椭圆弧
  - bool imagearc (resource image, int cx, int cy, int w, int h, int s, int e, int color)
  - imagearc() 以 *cx*, *cy*（图像左上角为 0, 0）为中心在 *image* 所代表的图像中画一个椭圆弧。*w* 和 *h* 分别指定了椭圆的宽度和高度，起始和结束点以 *s* 和 *e* 参数以角度指定。0° 位于三点钟位置，以顺时针方向绘画。
- imagefilledarc -- 画一椭圆弧且填充
  - bool imagefilledarc (resource image, int cx, int cy, int w, int h, int s, int e, int color, int style)
  - imagefilledarc() 在 *image* 所代表的图像中以 *cx*, *cy*（图像左上角为 0, 0）画一椭圆弧。如果成功则返回 TRUE，失败则返回 FALSE。*w* 和 *h* 分别指定了椭圆的宽和高，*s* 和 *e* 参数以角度指定了起始和结束点。*style* 可以是下列值按位或（OR）后的值：IMG_ARC_PIE、IMG_ARC_CHORD、IMG_ARC_NOFILL、IMG_ARC_EDGED。其中IMG_ARC_PIE 和 IMG_ARC_CHORD 是互斥的；IMG_ARC_CHORD 只是用直线连接了起始和结束点，IMG_ARC_PIE 则产生圆形边界（如果两个都用，IMG_ARC_CHORD 生效）。IMG_ARC_NOFILL 指明弧或弦只有轮廓，不填充。IMG_ARC_EDGED 指明用直线将起始和结束点与中心点相连，和 IMG_ARC_NOFILL 一起使用是画饼状图轮廓的好方法（而不用填充）。

3.6 在图像中绘制文字

- imagestring -- 水平地画一行字符串
  - 语法: bool imagestring (resource image, int font, int x, int y, string s, int col)
  - imagestring() 用 *col* 颜色将字符串 *s* 画到 *image* 所代表的图像的 *x*, *y* 坐标处（这是字符串左上角坐标，整幅图像的左上角为 0, 0）。如果 *font* 是 1, 2, 3, 4 或 5, 则使用内置字体。
- imagestringup -- 垂直地画一行字符串
  - 语法: bool imagestringup (resource image, int font, int x, int y, string s, int col)
  - imagestring()用 *col* 颜色将字符串 *s* 垂直地画到 *image* 所代表的图像的 *x*, *y* 座标处（图像的左上角为 0, 0）。如果 *font* 是 1, 2, 3, 4 或 5, 则使用内置字体。

- imagechar -- 水平地画一个字符
  - 语法: bool imagechar (resource image, int font, int x, int y, string c, int color)
  - imagechar() 将字符串 *c* 的第一个字符画在 *image* 指定的图像中，其左上角位于 *x*，*y*（图像左上角为 0，0），颜色为 *color*。如果 *font* 是 1，2，3，4 或 5，则使用内置的字体（更大的数字对应于更大的字体）。
- imagecharup -- 垂直地画一个字符
  - 语法: bool imagecharup (resource image, int font, int x, int y, string c, int color)
  - imagecharup() 将字符 *c* 垂直地画在 *image* 指定的图像上，位于 *x*，*y*（图像左上角为 0，0），颜色为 *color*。如果 *font* 为 1，2，3，4 或 5，则使用内置的字体。
- imagettftext -- 用 TrueType 字体向图像写入文本
  - 语法 : array imagettftext (resource image, float size, float angle, int x, int y, int color, string fontfile, string text)

```php
<?php
    $im = imagecreate(150,150);          //创建一个150*150的画布
    $bg = imagecolorallocate($im, 255, 255, 255);
      //设置画布的背景颜色为白色
    $black = imagecolorallocate($im, 0, 0, 0); //设置一个颜色变量为黑色
    $string="LAMPBrother";      //声明一个用于在图像中输出的字符串
    imageString($im, 3, 28, 70, $string, $black);
     //水平将字符串$string输出到图像中
    imageStringUp($im, 3, 59, 115, $string, $black);
    //垂直由下而上输出$string到图像中
    for($i=0,$j=strlen($string); $i<strlen($string); $i++,$j--){
          //使用循环单个字符输出到图像中
          imageChar($im, 3, 10*($i+1), 10*($i+2), $string[$i], $black);
          //向下倾斜输出每个字符
          imageCharUp($im, 3, 10*($i+1), 10*($j+2), $string[$i],
    $black);   //向上倾斜输出每个字符
    }
    header('Content-type: image/png');   //设置输出的头部标识符
    imagepng($im);                       //输出PNG格式的图片
?>
```

```php
<?php
    $im = imagecreatetruecolor(400, 30);      //创建400x300像素大小的画布
    $white = imagecolorallocate($im, 255, 255, 255);      //创建白色
    $grey = imagecolorallocate($im, 128, 128, 128);       //创建灰色
    $black = imagecolorallocate($im, 0, 0, 0);            //创建黑色
    imagefilledrectangle($im, 0, 0, 399, 29, $white);
     //输出一个使用白色填充的矩形作为背景
    //如果有中文输出，需要将其转码，转换为UTF-8的字符串才可以直接传递
    $text=iconv("GB2312", "UTF-8", "LAMP兄弟连--无兄弟，不编程！");
    $font = 'simsun.ttc';              //指定字体，将系统中与simsum.ttc对应
的字体复制到当前目录下
    imagettftext($im, 20, 0, 12, 21, $grey, $font, $text);    //输出一个
灰色的字符串作为阴影
    imagettftext($im, 20, 0, 10, 20, $black, $font, $text);   //在阴影之
上输出一个黑色的字符串
    header("Content-type: image/png");//通知浏览器将输出格式为PNG的图像
    imagepng($im);                          //向浏览器中输出PNG格式的图像
    imagedestroy($im);                      //销毁资源，释放内存占用的空间
?>
```

3.7 在PHP中实现验证码类的设计

谢谢！