**一个显示效果非常不错的PHP****错误、异常处理类**

一、效果图：

|  |
| --- |
| http://files.jb51.net/file_images/article/201403/2014321101634983.png?2014221101651 |
|  |

二、实现代码

复制代码代码如下:

<?**php**

// 自定义异常函数  
set\_exception\_handler('handle\_exception');

// 自定义**错误**函数  
set\_error\_handler('handle\_error');

/\*\*  
 \* 异常处理  
 \*  
 \* @param mixed $exception 异常对象  
 \* @author blog.snsgou.com  
 \*/  
function handle\_exception($exception) {  
 Error::exceptionError($exception);  
}

/\*\*  
 \* **错误**处理  
 \*  
 \* @param string $errNo **错误**代码  
 \* @param string $errStr **错误**信息  
 \* @param string $errFile 出错文件  
 \* @param string $errLine 出错行  
 \* @author blog.snsgou.com  
 \*/  
function handle\_error($errNo, $errStr, $errFile, $errLine) {  
 if ($errNo) {  
  Error::systemError($errStr, false, true, false);  
 }  
}

/\*\*  
 \* 系统**错误**处理  
 \*  
 \* @author blog.snsgou.com  
 \*/  
class Error {

 public static function systemError($message, $show = true, $save = true, $halt = true) {

  list($showTrace, $logTrace) = self::debugBacktrace();

  if ($save) {  
   $messageSave = '<b>' . $message . '</b><br /><b>**PHP**:</b>' . $logTrace;  
   self::writeErrorLog($messageSave);  
  }

  if ($show) {  
   self::showError('system', "<li>$message</li>", $showTrace, 0);  
  }

  if ($halt) {  
   exit();  
  } else {  
   return $message;  
  }  
 }

 /\*\*  
  \* 代码执行过程回溯信息  
  \*  
  \* @static  
  \* @access public  
  \*/  
 public static function debugBacktrace() {  
  $skipFunc[] = 'Error->debugBacktrace';

  $show = $log = '';  
  $debugBacktrace = debug\_backtrace();  
  ksort($debugBacktrace);  
  foreach ($debugBacktrace as $k => $error) {  
   if (!isset($error['file'])) {  
    // 利用反射API来获取方法/函数所在的文件和行数  
    try {  
     if (isset($error['class'])) {  
      $reflection = new ReflectionMethod($error['class'], $error['function']);  
     } else {  
      $reflection = new ReflectionFunction($error['function']);  
     }  
     $error['file'] = $reflection->getFileName();  
     $error['line'] = $reflection->getStartLine();  
    } catch (Exception $e) {  
     continue;  
    }  
   }

   $file = str\_replace(SITE\_PATH, '', $error['file']);  
   $func = isset($error['class']) ? $error['class'] : '';  
   $func .= isset($error['type']) ? $error['type'] : '';  
   $func .= isset($error['function']) ? $error['function'] : '';  
   if (in\_array($func, $skipFunc)) {  
    break;  
   }  
   $error['line'] = sprintf('%04d', $error['line']);

   $show .= '<li>[Line: ' . $error['line'] . ']' . $file . '(' . $func . ')</li>';  
   $log .= !empty($log) ? ' -> ' : '';  
   $log .= $file . ':' . $error['line'];  
  }  
  return array($show, $log);  
 }

 /\*\*  
  \* 异常处理  
  \*  
  \* @static  
  \* @access public  
  \* @param mixed $exception  
  \*/  
 public static function exceptionError($exception) {  
  if ($exception instanceof DbException) {  
   $type = 'db';  
  } else {  
   $type = 'system';  
  }  
  if ($type == 'db') {  
   $errorMsg = '(' . $exception->getCode() . ') ';  
   $errorMsg .= self::sqlClear($exception->getMessage(), $exception->getDbConfig());  
   if ($exception->getSql()) {  
    $errorMsg .= '<div class="sql">';  
    $errorMsg .= self::sqlClear($exception->getSql(), $exception->getDbConfig());  
    $errorMsg .= '</div>';  
   }  
  } else {  
   $errorMsg = $exception->getMessage();  
  }  
  $trace = $exception->getTrace();  
  krsort($trace);  
  $trace[] = array('file' => $exception->getFile(), 'line' => $exception->getLine(), 'function' => 'break');  
  $phpMsg = array();  
  foreach ($trace as $error) {  
   if (!empty($error['function'])) {  
    $fun = '';  
    if (!empty($error['class'])) {  
     $fun .= $error['class'] . $error['type'];  
    }  
    $fun .= $error['function'] . '(';  
    if (!empty($error['args'])) {  
     $mark = '';  
     foreach ($error['args'] as $arg) {  
      $fun .= $mark;  
      if (is\_array($arg)) {  
       $fun .= 'Array';  
      } elseif (is\_bool($arg)) {  
       $fun .= $arg ? 'true' : 'false';  
      } elseif (is\_int($arg)) {  
       $fun .= (defined('SITE\_DEBUG') && SITE\_DEBUG) ? $arg : '%d';  
      } elseif (is\_float($arg)) {  
       $fun .= (defined('SITE\_DEBUG') && SITE\_DEBUG) ? $arg : '%f';  
      } else {  
       $fun .= (defined('SITE\_DEBUG') && SITE\_DEBUG) ? '\'' . htmlspecialchars(substr(self::clear($arg), 0, 10)) . (strlen($arg) > 10 ? ' ...' : '') . '\'' : '%s';  
      }  
      $mark = ', ';  
     }  
    }  
    $fun .= ')';  
    $error['function'] = $fun;  
   }  
   if (!isset($error['line'])) {  
    continue;  
   }  
   $phpMsg[] = array('file' => str\_replace(array(SITE\_PATH, '\\'), array('', '/'), $error['file']), 'line' => $error['line'], 'function' => $error['function']);  
  }  
  self::showError($type, $errorMsg, $phpMsg);  
  exit();  
 }

 /\*\*  
  \* 记录**错误**日志  
  \*  
  \* @static  
  \* @access public  
  \* @param string $message  
  \*/  
 public static function writeErrorLog($message) {

  return false; // 暂时不写入

  $message = self::clear($message);  
  $time = time();  
  $file = LOG\_PATH . '/' . date('Y.m.d') . '\_errorlog.**php**';  
  $hash = md5($message);

  $userId = 0;  
  $ip = get\_client\_ip();

  $user = '<b>User:</b> userId=' . intval($userId) . '; IP=' . $ip . '; RIP:' . $\_SERVER['REMOTE\_ADDR'];  
  $uri = 'Request: ' . htmlspecialchars(self::clear($\_SERVER['REQUEST\_URI']));  
  $message = "<?**php** exit;?>\t{$time}\t$message\t$hash\t$user $uri\n";

  // 判断该$message是否在时间间隔$maxtime内已记录过，有，则不用再记录了  
  if (is\_file($file)) {  
   $fp = @fopen($file, 'rb');  
   $lastlen = 50000;  // 读取最后的 $lastlen 长度字节内容  
   $maxtime = 60 \* 10;  // 时间间隔：10分钟  
   $offset = filesize($file) - $lastlen;  
   if ($offset > 0) {  
    fseek($fp, $offset);  
   }  
   if ($data = fread($fp, $lastlen)) {  
    $array = explode("\n", $data);  
    if (is\_array($array))  
     foreach ($array as $key => $val) {  
      $row = explode("\t", $val);  
      if ($row[0] != '<?**php** exit;?>') {  
       continue;  
      }  
      if ($row[3] == $hash && ($row[1] > $time - $maxtime)) {  
       return;  
      }  
     }  
   }  
  }

  error\_log($message, 3, $file);  
 }

 /\*\*  
  \* 清除文本部分字符  
  \*  
  \* @param string $message  
  \*/  
 public static function clear($message) {  
  return str\_replace(array("\t", "\r", "\n"), " ", $message);  
 }

 /\*\*  
  \* sql语句字符清理  
  \*  
  \* @static  
  \* @access public  
  \* @param string $message  
  \* @param string $dbConfig  
  \*/  
 public static function sqlClear($message, $dbConfig) {  
  $message = self::clear($message);  
  if (!(defined('SITE\_DEBUG') && SITE\_DEBUG)) {  
   $message = str\_replace($dbConfig['database'], '\*\*\*', $message);  
   //$message = str\_replace($dbConfig['prefix'], '\*\*\*', $message);  
   $message = str\_replace(C('DB\_PREFIX'), '\*\*\*', $message);  
  }  
  $message = htmlspecialchars($message);  
  return $message;  
 }

 /\*\*  
  \* 显示**错误**  
  \*  
  \* @static  
  \* @access public  
  \* @param string $type **错误类**型 db,system  
  \* @param string $errorMsg  
  \* @param string $phpMsg  
  \*/  
 public static function showError($type, $errorMsg, $phpMsg = '') {  
  global $\_G;

  $errorMsg = str\_replace(SITE\_PATH, '', $errorMsg);  
  ob\_end\_clean();  
  $host = $\_SERVER['HTTP\_HOST'];  
  $title = $type == 'db' ? 'Database' : 'System';  
  echo <<<EOT  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html>  
<head>  
 <title>$host - $title Error</title>  
 <meta http-equiv="Content-Type" content="text/html; charset={$\_G['config']['output']['charset']}" />  
 <meta name="ROBOTS" content="NOINDEX,NOFOLLOW,NOARCHIVE" />  
 <style type="text/css">  
 <!--  
 body { background-color: white; color: black; font: 9pt/11pt verdana, arial, sans-serif;}  
 #container {margin: 10px;}  
 #message {width: 1024px; color: black;}  
 .red {color: red;}  
 a:link {font: 9pt/11pt verdana, arial, sans-serif; color: red;}  
 a:visited {font: 9pt/11pt verdana, arial, sans-serif; color: #4e4e4e;}  
 h1 {color: #FF0000; font: 18pt "Verdana"; margin-bottom: 0.5em;}  
 .bg1 {background-color: #FFFFCC;}  
 .bg2 {background-color: #EEEEEE;}  
 .table {background: #AAAAAA; font: 11pt Menlo,Consolas,"Lucida Console"}  
 .info {  
  background: none repeat scroll 0 0 #F3F3F3;  
  border: 0px solid #aaaaaa;  
  border-radius: 10px 10px 10px 10px;  
  color: #000000;  
  font-size: 11pt;  
  line-height: 160%;  
  margin-bottom: 1em;  
  padding: 1em;  
 }

 .help {  
  background: #F3F3F3;  
  border-radius: 10px 10px 10px 10px;  
  font: 12px verdana, arial, sans-serif;  
  text-align: center;  
  line-height: 160%;  
  padding: 1em;  
 }

 .sql {  
  background: none repeat scroll 0 0 #FFFFCC;  
  border: 1px solid #aaaaaa;  
  color: #000000;  
  font: arial, sans-serif;  
  font-size: 9pt;  
  line-height: 160%;  
  margin-top: 1em;  
  padding: 4px;  
 }  
 -->  
 </style>  
</head>  
<body>  
<div id="container">  
<h1>$title Error</h1>  
<div class='info'>$errorMsg</div>  
EOT;  
  if (!empty($phpMsg)) {  
   echo '<div class="info">';  
   echo '<p><strong>**PHP** Debug</strong></p>';  
   echo '<table cellpadding="5" cellspacing="1" width="100%" class="table"><tbody>';  
   if (is\_array($phpMsg)) {  
    echo '<tr class="bg2"><td>No.</td><td>File</td><td>Line</td><td>Code</td></tr>';  
    foreach ($phpMsg as $k => $msg) {  
     $k++;  
     echo '<tr class="bg1">';  
     echo '<td>' . $k . '</td>';  
     echo '<td>' . $msg['file'] . '</td>';  
     echo '<td>' . $msg['line'] . '</td>';  
     echo '<td>' . $msg['function'] . '</td>';  
     echo '</tr>';  
    }  
   } else {  
    echo '<tr><td><ul>' . $phpMsg . '</ul></td></tr>';  
   }  
   echo '</tbody></table></div>';  
  }  
  echo <<<EOT  
</div>  
</body>  
</html>  
EOT;  
  exit();  
 }  
}

/\*\*  
 \* DB异常**类**  
 \*  
 \* @author blog.snsgou.com  
 \*/  
class DbException extends Exception {

 protected $sql;  
 protected $dbConfig; // 当前数据库配置信息

 public function \_\_construct($message, $code = 0, $sql = '', $dbConfig = array()) {  
  $this->sql = $sql;  
  $this->dbConfig = $dbConfig;  
  parent::\_\_construct($message, $code);  
 }

 public function getSql() {  
  return $this->sql;  
 }

 public function getDbConfig() {  
  return $this->dbConfig;  
 }  
}