Log de errores por correo electrónico.

A todos nos preocupa, cuando desarrollamos una aplicación, que se produzcan errores. Y no hablamos de errores de sintaxis, como un final de línea, o unas comillas mal colocadas. Esos son, en su mayoría errores tipográficos durante la codificación, y son (casi siempre) muy fáciles de localizar y corregir.
Son mucho más difíciles de encontrar errores de planteamiento, como, por ejemplo, cuando tratamos de usar una variable no declarada. Suponiendo que logremos detectar dónde se trata de emplear, aún tenemos que saber dónde tendríamos que haberla declarado, y con qué valor. Y la cosa se complica más aún si estamos trabajando en una aplicación en arquitectura cliente-servidor, y la tenemos ya en producción. En nuestros servidores de desarrollo podemos tener activado el volcado de errores en pantalla, pero en producción no lo tendremos nunca. Simplemente, la aplicación no funciona, o "se cuelga", pero no nos dice por qué. Y si vas a decirme que no se debe poner nunca una aplicación en producción sin haberla testeado hasta la saciedad, te diré que sí, que eso suena muy bonito pero, en el mundo real, con mucha frecuencia el encargado de testearla se limita a echar una miradita rápida por encima, no sabe ni lo que tiene que comprobar, y te hacen ponerla en producción a toda prisa porque te la han encargado urgente, cuando el plazo de entrega ya ha vencido. A todos nos han hecho esas jugadas.
En estos casos, cualquier herramienta que te ayude a identificar y localizar posibles errores en tiempo de ejecución es una ayuda inestimable.

RECIBIR LOS ERRORES POR CORREO ELECTRÓNICO
Ya que no podemos contar con el volcado de errores en pantalla, es una buena idea que, cada vez que se genere un error en tiempo de ejecución, podamos recibir un correo electrónico con el máximo de detalles posibles. Y eso es lo que vamos a ver en este artículo. Se trata de un código que nos permite que la aplicación nos envíe un correo electrónico cada vez que se produce un comportamiento agnogénico imprevisto.
Lo primero que tenemos que hacer es determinar la ubicación, en nuestra aplicación, en la que vamos a colocar nuestro "chivato". Dado que debe estar disponible para toda la aplicación, es interesante colocarlo en algún script que sepamos que se va a a cargar siempre, desde cualquier parte. Un buen lugar para colocarlo es el script donde tengamos la conexión a base de datos, por ejemplo. Ese suele ser único para toda la aplicación, y se llama desde todos los procesos, ya que cualquier cosa que el usuario haga con la aplicación requerirá, antes o después, acceder a la base de datos.
Y, una vez determinada la ubicación correcta, pasamos a colocar nuestro testigo. Es muy simple, son muy pocas líneas, y ayuda muchísimo a detectar y localizar fallos.
EL TESTIGO]
Básicamente, el testigo es un error handler que, cuando se dispara, recaba datos del error que lo ha disparado, el script donde se ha producido, y otros datos adicionales, y los envía a una dirección de correo electrónico que le indiquemos. Veamos el código.

```
/* Vamos a crear una función, a la que llamaremos controlDeErrores,
que se ocupará de "recoger" cada error que ocurra en tiempo de ejecución,
recopilar una serie de datos acerca del mismo, y enviarlos, en un correo
electrónico, a la dirección que le indiquemos.
Se enviará un correo electrónico por cada error que se detecte */

/* Lo primero que hacemos es comprobar si la función que nos va a servir de testigo existe ya,
para evitar redeclararla. Sólo nos faltaría que un mecanismo de log de errores produjera,
```

```
a su vez, nuevos errores. Eso ya no es un error; es un horror.*/
if (!function_exists("controlDeErrores")){
 /* La función de control de errores recoje varios datos acerca del error producido.
 Estos datos se almacenan en unos parámetros que usaremos dentro.
 Cuando se produce un error en tiempo de ejecución, este "lanza" los siguientes datos:
 - Número de error. Es un código interno generado por PHP.
 - Descripción del error. La descripción del error es creada por PHP, y corresponde con el
 mensaje que veríamos si tuviéramos activado el volcado de errores en pantalla.
 - Nombre del script donde se ha producido el error.
 - Línea en la que se ha producido el error.
  */
 function controlDeErrores($errno, $errstr, $errfile, $errline){
  /* Inicializazamos el buffer de salida (ob, output buffer).
  A partir de ese momento, todas las salidas generadas que, normalmente, se volcarían en
  pantalla, quedan almacenadas en el bauffer de salida, un área de memoria que no se visualiza,
  a menos que lo indiquemos específicamente. */
  ob_start();
  echo "<pre>";
  /* Haciendo un volcado de $GLOBALS obtenemos, además, información de variables superglobales,
  como $_GET y $_POST, muy útil para comprobar que las lamadas a un formulario se hacen
  correctamente. */
  var_dump($GLOBALS);
  /* Cerramos el buffer de salida pero, en lugar de volcarlo a la pantalla,
  volcamos su contenido a una variable. */
  $contenidoError = ob_get_clean();
  /* Almacenamos las variables que lanzó el error (y que, recordemos, han pasado como
  parámetros a esta función),
  así como el volcado del bufer, que contiene las variables globales, en un mensaje para enviar. */
  $mensajeParaEnviar = "Error: [$errno] $errstr $errfile $errline rn $contenidoError";
  /* La cadena de los errores se envia al correo que aparece en el tercer parámetro de la
  función error_log(), con el remite que aparece en el cuarto parámetro. */
  error_log($mensajeParaEnviar, 1, "depurador@dominio.es", "From: aplicacion@aplicacion.com");
 }
}

/* La siguiente línea es de vital importancia. Es lo que activa el manejador
de errores (una especie de EventListener orientado a los errores de ejecución)
y lo asocia a la función que hemos definido, de forma que esta se ejecute cada
vez que se produce un error.*/
set_error_handler("controlDeErrores");
```

El mecanismo de control de errores en sí es mucho más sencillo de lo que parece. Si lo observas, verás que la mayoría del código son comentarios que detallan, paso a paso, su funcionamiento.

REFERENCIAS EXTERNAS]

Aunque el mecanismo que te presento aquí funciona perfectamente, no está de más documentarse sobre algunas funciones de PHP que, fuera de este contexto, no se usan demasiado, pero que puede resultarnos muy útil conocer:

El Output Buffer. Es una interesante herramienta de PHP que nos permite almacenar salidas, y hacer un volcado en el momento deseado, redireccionándolo, además, si lo deseamos, a una variable. Puede leer sobre él en este enlace.

La función error_log(), que envía el error a donde le indiquemos. La documentación oficial está en este enlace.

El **EventListener** para los errores. La documentación oficial está en este enlace.