Errores y manejo de excepciones

PHP define una clasificación de los errores que se pueden producir en la ejecución de un programa y ofrece métodos para ajustar el tratamiento de los mismos. Para hacer referencia a cada uno de los niveles de error, PHP define una serie de constantes. Cada nivel se identifica por una constante.

P.ej.: la constante E\_NOTICE hace referencia a avisos que pueden indicar un error al ejecutar el guión, y la constante E\_ERROR engloba errores fatales que provocan que se interrumpa forzosamente la ejecución.

La configuración inicial de tratamiento de errores según su nivel se realiza en php.ini (fichero de configuración de PHP). Los parámetros que se pueden ajustar son:

**error\_reporting**. Indica qué tipos de errores se notificarán. Su valor se forma combinando las constantes anteriores. Su valor predeterminado es E\_ALL &~E\_NOTICE que indica que se notifiquen todos los errores (E\_ALL) salvo los avisos en tiempo de ejecución (E\_NOTICE).

**display\_errors.** En su valor por defecto (On), hace que los mensajes se envíen a la salida estándar (y por lo tanto se muestren en el navegador). Se debe desactivar (Off) en los servidores que se usan para producción.

Desde el código, se puede usar la función error\_reporting con las constantes anteriores para establecer el nivel de notificación en un momento determinado.

P.ej.: si en algún lugar figura una división en la que exista la posibilidad de que el divisor sea cero, cuando ello ocurra se obtendrá un mensaje de error en el navegador. Para evitarlo, se puede desactivar la notificación de errores de nivel E\_WARNING antes de la división y restaurarla a su valor normal a continuación:

error\_reporting(E\_ALL & ~E\_NOTICE & ~E\_WARNING);

$resultado = $dividendo / $divisor;

error\_reporting(E\_ALL & ~E\_NOTICE);

Al usar la función error\_reporting solo se controla qué tipo de errores va a

notificar PHP. Puede ser suficiente, pero para obtener más control sobre el proceso existe la posibilidad de reemplazar la gestión de los mismos. Es decir, es posible programar una función para que sea la que ejecuta PHP cada vez que se produce un error. El nombre de esa función se indica utilizando **set\_error\_handler**, y debe tener dos parámetros obligatorios (el nivel del error y el mensaje descriptivo) y hasta otros tres opcionales (el nombre del fichero en que se produce, el número de línea, y un volcado del estado de las variables en ese momento).

set\_error\_handler("miGestorDeErrores");

$resultado = $dividendo / $divisor;

restore\_error\_handler();

function miGestorDeErrores($nivel, $mensaje)

{

switch($nivel) {

case E\_WARNING:

echo "Error de tipo WARNING: $mensaje.<br />";

break;

default:

echo "Error de tipo no especificado: $mensaje.<br />";

}

}

La función restore\_error\_handler restaura el manejador de errores original de PHP (el que se estaba usando antes de la llamada a set\_error\_handler).

PHP utiliza un modelo de excepciones similar al de otros lenguajes:

* El código susceptible de producir algún error se introduce en un bloque try.
* Cuando se produce algún error, se lanza una excepción utilizando la instrucción throw.
* Después del bloque try debe haber como mínimo un bloque catch encargado de procesar el error.
* Si una vez acabado el bloque try no se ha lanzado ninguna excepción, se continúa con la ejecución en la línea siguiente al bloque o bloques catch.

P.ej.: para lanzar una excepción cuando se produce una división por cero se podría hacer:

try {

if ($divisor == 0)

throw new Exception("División por cero.");

$resultado = $dividendo / $divisor;

}

catch (Exception $e) {

echo "Se ha producido el siguiente error: ".$e->getMessage();

}

PHP ofrece una **clase base Exception** para utilizar como manejador de excepciones. Para lanzar una excepción no es necesario indicar ningún parámetro, aunque de forma opcional se puede pasar un mensaje de error y también un código de error.

Entre los métodos que se pueden usar con los objetos de la clase Exception están:

o **getMessage**. Devuelve el mensaje, en caso de que se haya puesto alguno.

o **getCode.** Devuelve el código de error si existe.

Las funciones internas de PHP y muchas extensiones como MySQLi usan el sistema de errores visto anteriormente. Solo las extensiones más modernas orientadas a objetos, como es el caso de PDO, utilizan este modelo de excepciones.

Concretamente, la clase PDO permite definir la fórmula que usará cuando se produzca un error, utilizando el atributo PDO::ATTR\_ERRMODE. Las posibilidades son:

* PDO::ERRMODE\_SILENT. No se hace nada cuando ocurre un error. Comportamiento por defecto.
* PDO::ERRMODE\_WARNING. Genera un error de tipo E\_WARNING cuando se produce un error.
* PDO::ERRMODE\_EXCEPTION. Cuando se produce un error lanza una excepción utilizando el manejador propio PDOException.

Es decir, para utilizar excepciones con la extensión PDO, se debe configurar la conexión haciendo:

$dwes->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

P.ej.: el siguiente código:

$dwes = new PDO("mysql:host=localhost; dbname=dwes", "dwes", "abc123.");

$dwes->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

try {

$sql = "SELECT \* FROM stox";

$result = $dwes->query($sql);

…

}

catch (PDOException $p) {

echo "Error ".$p->getMessage()."<br />";

}

Captura la excepción que lanza PDO debido a que la tabla no existe. El bloque catch muestra el siguiente mensaje:

Error SQLSTATE[42S02]: **Base table or view not found: 1146 Table 'dwes.stox' doesn't exist**