Excepciones y errores

El sistema de control de errores de PHP ha ido evolucionando a lo largo de las versiones. En el sistema básico, se generan errores de diferentes tipos, representados por un número. Por otro lado, desde PHP5 hay un sistema de excepciones similar al de Java y otros lenguajes utilizando bloques ***try/catch/finally***. Finalmente, en PHP7 aparecieron las excepciones de clase ***Error***.

Errores

En el sistema básico, ante determinadas condiciones, PHP genera un error asociado con un número y una constante predefinida.

Se puedo controlar cómo se comporta PHP ante los errores mediante algunas directivas del fichero *php.ini*:

- *error_reporting*: indica qué errores deben reportarse. Lo normal es utilizar E_ALL, es decir, todos.
- *display_errors*: señala si los mensajes de error deben aparecer en la salida del *script*. Esta opción es apropiada durante el desarrollo, pero no en producción.

- *log_errors*: indica si los mensajes de error deben almacenarse en un fichero. Es especialmente útil en producción, cuando no se muestran los errores en la salida.
    - *error_log*: si la directiva anterior está activada, es la ruta en la que se guardan los mensajes de error.

El valor de la directiva error_reporting es un número, pero para especificarlo lo habitual es utilizar las constantes predefinidas y el operador `or` a nivel de bit.

Código	Constante	Descripción
1	E_ERROR	Errores Fatales en tiempo de ejecución. Éstos indican errores que no se pueden recuperar, tales como un problema de asignación de memoria. La ejecución del script se interrumpe.
2	E_WARNING	Advertencias en tiempo de ejecución (errores no fatales). La ejecución del script no se interrumpe.
4	E_PARSE	Errores de análisis en tiempo de compilación. Los errores de análisis deberían ser generados únicamente por el analizador.
8	E_NOTICE	Avisos en tiempo de ejecución. Indican que el script encontró algo que podría señalar un error, pero que también podría ocurrir en el curso normal al ejecutar un script.
16	E_CORE_ERROR	Errores fatales que ocurren durante el arranque incial de PHP. Son como un **E_ERROR**, excepto que son generados por el núcleo de PHP.
32	E_CORE_WARNING	Advertencias (errores no fatales) que ocurren durante el arranque inicial de PHP. Son como un **E_WARNING**, excepto que son generados por el núcleo de PHP.
64	E_COMPILE_ERROR	Errores fatales en tiempo de compilación. Son como un **E_ERROR**, excepto que son generados por Motor de Script Zend.

128	E_COMPILE_WARNING	Advertencias en tiempo de compilación (errores no fatales). Son como un **E_WARNING**, excepto que son generados por Motor de Script Zend.
256	E_USER_ERROR	Mensaje de error generado por el usuario. Es como un **E_ERROR**, excepto que es generado por código de PHP mediante el uso de la función de PHP trigger_error().
512	E_USER_WARNING	Mensaje de advertencia generado por el usuario. Es como un **E_WARNING**, excepto que es generado por código de PHP mediante el uso de la función de PHP trigger_error().
1024	E_USER_NOTICE	Mensaje de aviso generado por el usuario. Es como un **E_NOTICE**, excepto que es generado por código de PHP mediante el uso de la función de PHP trigger_error().
2048	E_STRICT	Habilítelo para que PHP sugiera cambios en su código, lo que asegurará la mejor interoperabilidad y compatibilidad con versiones posteriores de PHP de su código.
4096	E_RECOVERABLE_ERROR	Error fatal capturable. Indica que ocurrió un error probablemente peligroso, pero no dejó al Motor en un estado inestable. Si no se captura el error mediante un gestor definido por el usuario (vea también set_error_handler()), la aplicación se abortará como si fuera un **E_ERROR**.
8192	E_DEPRECATED	Avisos en tiempo de ejecución. Habilítelo para recibir avisos sobre código que no funcionará en futuras versiones.
16384	E_USER_DEPRECATED	Mensajes de advertencia generados por el usuario. Son como un **E_DEPRECATED**, excepto que es generado por código de PHP mediante el uso de la función de PHP trigger_error().
32767	E_ALL	Todos los errores, advertencias y avisos.

Excepciones

Otra opción para indicar un error es lanzar una excepción. Para controlar las excepciones se utilizan bloques **try/catch/finally**, como en Java. Cuando se lanza una excepción y no es capturada por un bloque catch, la

ejecución del programa se detiene. Si es capturada, se ejecuta el código del bloque correspondiente.

En el siguiente ejemplo se utiliza una función que recibe dos argumentos, lanza una excepción si el segundo argumento es cero. Para ello se utiliza **throw**, que recibe como argumento un objeto de clase Exception o de alguna subclase.

```php
<?php

function dividir($a, $b) {
    if ($b==0) {
        throw new Exception('El segundo argumento es cero');
    }

    return $a/$b;
}

try {
    $resul1 = dividir(5, 0);
    echo "Resul 1 $resul1"."<br />";
}
catch(Exception e) {
    echo "Excepción: " . $e->getMessage() . "<br />";
}
finally {
    echo "Primer Finally";
}

try {
    $resul2 = dividir(7, 3);
    echo "Resul 2 $resul2"."<br />";
}
catch(Exception e) {
    echo "Excepción: " . $e->getMessage() . "<br />";
}
finally {
    echo "Segundo Finally";
}

?>
```

En la primera llamada a dividir(), línea 12, se produce una excepción. Por tanto, el resto del bloque *try* no se ejecuta. Sí se ejecutan el *catch* correspondiente y el *finally*.

En la segunda llamada a dividir(), línea 23, no se produce la excepción. En este caso se ejecuta el resto del *try* y a continuación el del bloque *finally* correspondiente. El bloque *catch* no se ejecuta.

Excepciones Error

En PHP7 aparecieron las excepciones de tipo **Error**. No heredan de la clase Exception, así que para capturarlas hay que usar:

```
catch (Error $e) {
...
}
```

O, alternativamente, la clase Throwable, de la que se derivan tanto Error como Exception:

```
catch (Throwable $e) {
...
}
```

En la siguiente tabla se muestran las excepciones Error predefinidas.

Nombre	Descripción	Hereda de
Error	Clase base para las excepciones Error	
ArithmeticError	Error en operaciones matemáticas.	Error
DivisionByZeroError	Intento de división por cero.	ArithmeticError
AssertionError	Ocurre cuando falla una llamada a assert().	Error
ParseError	Error al compilar.	Error
TypeError	Ocurre cuando una expresión no tiene el tipo de dato que se espera.	Error
ArgumentCountError	Ocurre al llamar a una función con menos argumentos de los necesarios.	TypeError