Unidad 4 - Errores y excepciones

Introducción

La depuración de errores es una tarea esencial en programación. En PHP, el sistema de control de errores ha evolucionado desde sus versiones iniciales:

- **PHP 5** introdujo el manejo de excepciones mediante la clase `Exception`, permitiendo capturar y gestionar errores de forma personalizada.
- **PHP 7** añadió la clase `Error`, extendiendo el manejo de errores internos de PHP como si fueran excepciones.

Tipos de errores en PHP

1. **Errores Críticos (`Fatal Error`)** → Detienen la ejecución del programa de manera inmediata.

   Ejemplo: Intentar llamar a una función no definida.

2. **Errores de Análisis (`Parse Error`)** → Detectados por el analizador de PHP antes de ejecutar el código. Comunes en errores de sintaxis.

   Ejemplo: Olvidar `$` en una variable o no cerrar un paréntesis.

3. **Advertencias (`Warning`)** → No detienen la ejecución, pero pueden causar problemas graves posteriormente.

   Ejemplo: Usar `include()` con un archivo inexistente.

4. **Notificaciones (`Notice`)** → Indican situaciones no críticas que podrían afectar el comportamiento del programa.

   Ejemplo: Usar una variable no inicializada.

Configuración del archivo `php.ini`

El comportamiento de PHP frente a errores se configura con las siguientes directivas:

- `error_reporting`: Define los tipos de errores a notificar.
  - `E_ALL` → Todos los errores.
  - `E_ALL & ~E_NOTICE` → Todos excepto los avisos.
- `display_errors`: Muestra errores en la salida estándar (activado por defecto). En entornos de producción suele desactivarse por seguridad.
- `log_errors`: Guarda errores en un archivo log (recomendado en producción si `display_errors` está desactivado).
- `error_log`: Especifica la ruta del archivo donde se registran los errores.

Control de errores

Funciones clave para manejar errores

- `error_reporting()` → Establece el nivel de notificación de errores dinámicamente.
- `set_error_handler()` → Permite definir un controlador personalizado para los errores.

```
function manejadorErrores($nivel, $mensaje) {
    switch ($nivel) {
        case E_WARNING:
            echo "Advertencia: $mensaje";
            break;
        case E_NOTICE:
            echo "Aviso: $mensaje";
            break;
        default:
            echo "Error: $mensaje";
    }
    error_log($mensaje);
}
set_error_handler('manejadorErrores');
```

- `restore_error_handler()` → Restaura el manejador de errores interno de PHP.
- `trigger_error()` → Genera errores personalizados.

Manejo de excepciones

Estructura básica:

1. `try` → Bloque que contiene el código propenso a errores.
2. `throw` → Lanza una excepción en caso de error.
3. `catch` → Captura y gestiona la excepción lanzada.
4. `finally` (Opcional) → Bloque que se ejecuta siempre, ideal para tareas de limpieza.

```
try {
    $dividendo = 0;
    if ($dividendo == 0) {
        throw new Exception("División por cero.");
    }
    $resultado = 10 / $dividendo;
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
} finally {
    echo "Ejecución finalizada.";
}
```

Personalización de excepciones

Las excepciones pueden personalizarse creando clases que heredan de `Exception`:

```
class MiExcepcion extends Exception {
    public function mensajePersonalizado() {
        return "Error en la línea {$this->getLine()}: {$this->getMessage
```

```
()}";
    }
}

try {
    throw new MiExcepcion("Formato inválido");
} catch (MiExcepcion $e) {
    echo $e->mensajePersonalizado();
}
```

Errores Manejables

Los errores manejables permiten reemplazar el formato nativo de PHP mediante controladores personalizados.

Tipos de errores manejables:

- `E_WARNING`, `E_NOTICE`
- `E_USER_ERROR`, `E_USER_WARNING`, `E_USER_NOTICE`
- `E_RECOVERABLE_ERROR`

Clases relacionadas con errores y excepciones

PHP proporciona clases predefinidas para la gestión de errores y excepciones:

- **`Exception`** → Base para excepciones personalizadas.
- **`Error`** → Base para tratar errores nativos de PHP como `excepciones.Subclases`:
  - `TypeError`
  - `DivisionByZeroError`
  - `AssertionError`

Conclusiones

- Todo programa debe incluir un sistema de control de errores y excepciones.
- El manejo de errores y excepciones contribuye a:
  - **Seguridad** → Ocultar detalles técnicos al usuario final.
  - **Mantenimiento** → Registrar errores en logs para análisis posterior.
- Aunque similares, los errores y excepciones tienen enfoques distintos:
  - Los errores son eventos inesperados gestionados por PHP.
  - Las excepciones permiten controlar el flujo del programa cuando ocurren situaciones excepcionales.