UD2. Introducción al lenguaje PHP

Desarrollo Web en Entorno Servidor

Sistema de errores (II)
Clases y objetos

Errores y excepciones

excepciones.php

El sistema de errores de PHP ha ido evolucionando:

1. En el **sistema básico**: los errores se identifican con un código (**una constante**).
2. Desde **PHP 5**: se añaden las **excepciones** y el bloque **`try-catch-finally`**.
3. Desde **PHP 7**: se añaden las **excepciones de clase Error**.

Errores: sistema básico

En el **sistema básico** ante algunas condiciones se genera un error (ej. variable no inicializada).

- La mayoría de funciones internas de PHP utilizan el sistema básico.

Códigos de errores y descripción: https://www.php.net/manual/es/errorfunc.constants.php

Errores: sistema básico

¿Cómo los capturo entonces?

Opción A)

Se puede configurar cómo funciona PHP cuando encuentra un error tocando el `php.ini`:

- `error_reporting`: qué errores se deben reportar. Lo normal: `E_ALL`
- `display_errors`: cuáles se imprimen.
- `log_errors`: cuáles se deben guardar en un fichero.

Opción B)

Hago lo anterior en ejecución con `set_error_handler("nombre_función");`

Errores: sistema básico

Opción B)

Hago lo anterior en ejecución con `set_error_handler("nombre_función");`

```
<?php
	function manejadorErr($errno, $str, $file, $line) {
		echo "Ocurrió un error " . $errno;
	}
	set_error_handler("manejadorErr");
	$a = $b;
```

Errores: PHP 7

Existen otros errores a partir de PHP 7 que se lanzan como una excepción.

- https://www.php.net/manual/es/language.errors.php7.php
- Para capturarlos: try-catch(Error $e)-finally;
- O bien: try-catch(Throwable $e)-finally;

Excepciones

excepciones.php

- Lanzamiento con `throw Exception(string);`
- Captura con `try-catch(Exception $e)-finally.`

Excepciones

excepciones.php

Captura:

```
try {
   instrucciones;
} catch (Exception e) {
   instrucciones;
} finally {
   instrucciones;
}
```

Lanzamiento:

```
throw new Exception("Mensaje");
```

Clases

PersonayCliente.php

- PHP tiene soporte completo para la programación orientada a objetos.
- Permite definir clases, herencia, interfaces y los demás elementos habituales.
- Para declarar una clase se utiliza la palabra reservada `class`.

```
class Clase{
      private $att1 = 10;          // con valor por defecto
      private $atr2;               // sin valor por defecto
      private static $atr3 = 0;    // estático
      …
      public function metodo() { … }
}
```

Para crear un objeto:

```
$var = new Clase();
```

Clases

PersonayCliente.php

Acceso a función o método:

```
$objeto->atributo;
$objeto->metodo($argumento);
```

Constructor:

```
function __construct($valor1, $valor2){
    $this->$atributo = $valor1;
    …
}
```

Modificadores de **visibilidad**:

- `public`: el elemento se puede usar **fuera y dentro de la clase**.
- `private`: el elemento solo puede usarse **dentro de la clase**.
- `protected`: el elemento puede usarse **dentro de la clase y las derivadas**.

Clases

PersonayCliente.php

Método mágico para echo:

```
function __toString()
```

Herencia:

Para heredar una clase de otra: `class Hijo` **`extends`** `Padre`