Excepciones en PHP

Heike Bonilla Redondo

1. Excepciones¶

La gestión de excepciones forma parte desde PHP 5. Su funcionamiento es similar a *Java*, haciendo uso de un bloque try / catch. Si detectamos una situación anómala y queremos lanzar una excepción, deberemos realizar throw new Exception (adjuntando el mensaje que lo ha provocado).

```
<?php
$divisor=0;
try {
    if ($divisor == 0) {
        throw new Exception("División por cero.");
    }
    $resultado = $dividendo / $divisor;
} catch (Exception $e) {
    echo "Se ha producido el siguiente error: ".$e->getMessage();
}
?>
```

La clase Exception es la clase padre de todas las excepciones. Su constructor recibe mensaje[,codigoError][,excepcionPrevia].

```
public Exception::__construct(string $message = "", int $code = 0,
?Throwable $previous = null)
```

A partir de un objeto Exception, podemos acceder a los siguientes métodos

- Exception::getMessage — Obtiene el mensaje de Excepción
- Exception::getPrevious — Devuelve la Throwable anterior
- Exception::getCode — Obtiene el código de una excepción
- Exception::getFile — Obtiene el fichero en el que se creó la excepción
- Exception::getLine — Obtiene la línea en el que se creó la excepción
- Exception::getTrace — Obtiene la traza de la pila
- Exception::__toString — Representación de la excepción en formato cadena

El propio lenguaje ofrece un conjunto de excepciones ya definidas, las cuales podemos capturar (y lanzar desde PHP 7). Se recomienda su consulta en la documentación oficial.

Creando excepciones¶

Para crear una excepción, la forma más corta es crear una clase que únicamente herede de Exception.

<?php

```
class HolaExcepcion extends Exception {}
```

Si queremos, y es recomendable dependiendo de los requisitos, podemos sobrecargar los métodos mágicos, por ejemplo, sobrecargando el constructor y llamando al constructor del padre, o rescribir el método __toString para cambiar su mensaje:

```php
<?php

class MiExcepcion extends Exception {
    public function __construct($msj, $codigo = 0, Exception $previa = null)
{
        // código propio
        parent::__construct($msj, $codigo, $previa);
    }
    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }
    public function miFuncion() {
        echo "Una función personalizada para este tipo de excepción\n";
    }
}

//Ejemplo de uso de mi nueva clase de excepción
try {
    if (true) {
         throw new MiExcepcion("Esta es mi nueva clase de excepción.");
    }

} catch (MiExcepcion $me) {
    echo "Mensaje: ".$me->getMessage();
    echo"<br>";
    echo $me;
}

?>
```

Excepciones múltiples¶

Se pueden usar excepciones múltiples para comprobar diferentes condiciones. A la hora de capturarlas, se hace de más específica a más general.

```php
<?php

class MiExcepcion extends Exception {
    public function __construct($msj, $codigo = 0, Exception $previa = null)
{
        // código propio
        parent::__construct($msj, $codigo, $previa);
    }
    public function __toString() {
        return __CLASS__ . ": [{$this->code}]: {$this->message}\n";
    }
    public function miFuncion() {
        echo "Una función personalizada para este tipo de excepción\n";
    }
}

//Ejemplo de uso de mi nueva clase de excepción

$email = "ejemplo@ejemplo.com";
try {
    // Comprueba si el email es válido
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
```

```
        throw new MiExcepcion($email);
    }
    // Comprueba la palabra ejemplo en la dirección email
    if(strpos($email, "ejemplo") !== FALSE) {
        throw new Exception("$email es un email de ejemplo no válido");
    }
} catch (MiExcepcion $e) {
    echo $e->miFuncion();
} catch(Exception $e) {
    echo $e->getMessage();
}

?>
```

Autoevaluación

¿Qué pasaría al ejectuar el siguiente código?

```
<?php
class MainException extends Exception {}
class SubException extends MainException {}

try {
    throw new SubException("Lanzada SubException");
} catch (MainException $e) {
    echo "Capturada MainException " . $e->getMessage()."<br>";
} catch (SubException $e) {
    echo "Capturada SubException " . $e->getMessage()."<br>";
} catch (Exception $e) {
    echo "Capturada Exception " . $e->getMessage()."<br>";
}
?>
```

Si en el mismo catch queremos capturar varias excepciones, hemos de utilizar el operador |:

```
<?php
class MainException extends Exception {}
class SubException extends MainException {}

try {
    throw new SubException("Lanzada SubException");
} catch (MainException | SubException $e ) {
    echo "Capturada Exception " . $e->getMessage();
}

?>
```

Desde PHP 7, existe el tipo Throwable, el cual es un interfaz que implementan tanto los errores como las excepciones, y nos permite capturar los dos tipos a la vez:

```
<?php
try {
    // tu codigo
} catch (Throwable $e) {
```

```
    echo 'Forma de capturar errores y excepciones a la vez';
}
```

Si sólo queremos capturar los errores fatales, podemos hacer uso de la clase Error:

```
<?php
try {
    // Genera una notificación que no se captura
    echo $variableNoAsignada;
    // Error fatal que se captura
    funcionQueNoExiste();
} catch (Error $e) {
    echo "Error capturado: " . $e->getMessage();
}
```