HANDLING EXCEPTIONS AND ERRORS IN PHP

ERROR HANDLING IS AN ESSENTIAL PART OF DEVELOPING AN APPLICATION. PHP PROVIDES A NUMBER OF TOOLS THAT WE CAN USE TO HANDLE ANY KIND OF ERRORS.

IN THIS UNIT WE WILL SEE HOW TO DO IT AT TWO LEVELS:

- CATCHING AND HANDLING THROWABLE ERRORS
- CONFIGURING PHP TO HANDLE OTHER TYPES OF ERRORS

HANDLING EXCEPTIONS AND ERRORS IS SLIGHTLY DIFFERENT IN PHP7 COMPARED TO PREVIOUS VERSIONS. WE WILL GO THROUGH THOSE DIFFERENCES IN THIS LESSON.

EXCEPTIONS

A PHP EXCEPTION HAPPENS WHEN THE APPLICATION TRIES TO PERFORM A TASK AND IT'S UNABLE TO DO IT.

AN EXCEPTION STOPS THE EXECUTION UNLESS WE CATCH IT AND HANDLE IT.

BY CATCHING EXCEPTIONS WE CAN:

- AVOID SHOWING UNDESIRED ERROR MESSAGES TO THE FINAL USER
- PREVENT THE APPLICATION TO BE SUDDENLY HALTED

HOW TO HANDLE EXCEPTIONS

IN PHP5 THE EXCEPTIONS WOULD BE HANDLED BY USING THE `Exception` CLASS AND THE STRUCTURE `try...catch`

```
try
    {
        ...
    }
catch(Exception $e)
    {
        echo $e->getMessage();
    }
```

IN PHP5 SOME INTERNAL ERRORS CAN NOT BE HANDLED BY USING THE `Exception` CLASS. IN PHP7 WE HAVE THE CLASS `Throwable` INSTEAD. IT COVERS BOTH EXCEPTIONS AND INTERNAL ERRORS.

HOW TO HANDLE EXCEPTIONS

JUST IN CASE WE ARE NOT SURE IF THE SERVER SUPPORTS PHP5 OR PHP7, WE CAN INCLUDE BOTH CLAUSES IN OUR `try...catch` BLOCK.

```
try
{
   // Code that may cause an Exception or Error.
}
catch (Throwable $t)
{
   // Executed only in PHP 7, will not match in PHP 5
}
catch (Exception $e)
{
   // Executed only in PHP 5, will not be reached in PHP 7
}
```

EXCEPTIONS

LET'S SEE AN EXAMPLE. IF WE RUN THIS CODE

```
<body>
   <?php
    $number=10;
    $anverseNumber=1/$number;
    echo "<h2>The inverse of $number is $anverseNumber</h2>";
   ?>
</body>
```

THE OUTPUT WILL BE SOMETHING LIKE THIS:

The inverse of 10 is 0.1

EXCEPTIONS

BUT, IF WE CHANGE THE VALUE OF `$number` TO 0:

```
<body>
   <?php
    $number=0;
    $anverseNumber=1/$number;
    echo "<h2>The inverse of $number is $anverseNumber</h2>";
   ?>
</body>
```

WE WILL GET AN ERROR MESSAGE:

Fatal error: Uncaught DivisionByZeroError: Division by zero

EXCEPTIONS

WE CAN HANDLE THE EXCEPTION BY USING THE `try...catch` BLOCK

```
<?php
   $number=0;
   try {
      $anverseNumber=1/$number;
      echo "<h2>The inverse of $number is $anverseNumber</h2>";
   }
   catch(Throwable $t) {
         echo "An error happened";
   }
?>
```

IF AN ERROR OCCURS WITHIN THE TRY SECTION, THE EXECUTION WILL NOT BE HALTED. IT WILL BE REDIRECTED TO THE CATCH SECTION INSTEAD.

WE WILL GET THIS OUTPUT:

An error happened

EXCEPTIONS

YOU MIGHT WANT TO KNOW MORE ABOUT THE EXCEPTION. BOTH `Throwable` AND `Exception` PROVIDE YOU WITH SOME METHODS THAT HELP:

```
getMessage(); // returns the name of the exception

getCode();    // returns the code of the exception

getFile();    // returns the name of the file where the exception happened

getLine();    // returns the line in the file where the exception happened

__toString(); // returns the exception in a String format
```

EXCEPTIONS

```
<?php
  $number=0;
  try {
     $anverseNumber=1/$number;
     echo "<h2>The inverse of $number is $anverseNumber</h2>";
  }
  catch(Throwable $t) {
     echo "An error {$t->getMessage()} happened<br/>";
     echo "In line {$t->getLine()} of file {$t->getFile()}<br/>";
  }
?>
```

An error Division by zero happened
In line 17 of file C:\xampp\htdocs\ProvesPHP\exceptions\exceptions01.php

EXCEPTIONS

WE CAN HANDLE SPECIFIC EXCEPTIONS BY REPLACING THE GENERAL CLAUSE `Throwable` WITH THE NAME OF THE EXCEPTION IN THE CATCH SECTION

```
catch (DivisionByZeroError $t)
```

THUS, WE CAN HANDLE DIFFERENT EXCEPTIONS WITHIN THE SAME `try...catch` BLOCK

```
try {
    ...
}
catch (DivisionByZeroError $t) {
    ...
}
catch (ArithmeticError $e) {
    …
}
```

WARNINGS

SOME ERRORS ARE NOT THROWABLE. LET'S TAKE A LOOK AT THIS EXAMPLE

```
<?php
  function anverse($number) {
    $anverseNumber = 1 / $number;
    return $anverseumber;
  }
  try {
    $number=10;
    echo "<h2>The inverse of $number is ".anverse($number)."</h2>";
  }
  catch (Throwable $t) {
    echo "An error {$t->getMessage()} happened<br/>";
  }
?>
```

WHEN WE RUN THE SCRIPT, WE WILL GET A WARNING:

Warning: Undefined variable $anverseumber in **C:\xampp\htdocs\ProvesPHP\exceptions\exceptions01.php** on line **17**

The inverse of 10 is

WARNINGS

A WARNING IS A NON-FATAL ERROR THAT SHOWS A MESSAGE, BUT DOESN'T HALT THE APPLICATION.

A WARNING CAN'T GET HANDLED BY THE `try...catch` STRUCTURE.

WARNINGS CAN BE DISABLED IN THE php.ini CONFIGURATION FILE, OR BY USING THE FUNCTION `error_reporting(E_ERROR)`. HOWEVER, HANDLING THEM IS A BETTER PRACTICE IN DEVELOPING MODE.

WARNINGS

WE CAN HANDLE WARNINGS BY TRANSFORMING THEM INTO EXCEPTIONS WITH OUR OWN ERROR HANDLING FUNCTION.

```
function handlingErrors($eLevel, $eMessage, $eFile, $eLine) {
  throw new Exception("Error ".$eMessage." in line ".$eLine." of ".
$eFile);
}
```

AND THEN WE DEFINE WHAT OUR ERROR HANDLING FUNCTION WILL BE

```
set_error_handler("handlingErrors");
```

WE SHOULD RESTORE THE AUTOMATIC ERROR HANDLER AT THE END OF THE SCRIPT

```
restore_error_handler();
```

WARNINGS

```
<?php
  function handlingErrors($eLevel, $eMessage, $eFile, $eLine) {
   throw new Exception("Error ".$eMessage." in line ".$eLine." of
   ".$eFile); // both warnings and exceptions will be thrown
  }
  function anverse($number) {
      $anverseNumber = 1 / $number;
     return $anverseumber;
  }
  set_error_handler("handlingErrors");
  try {
    $number=10;
    echo "<h2>The inverse of $number is ".anverse($number)."</h2>";
  }
  catch (Throwable $t) {
     echo "An error {$t->getMessage()} happened<br/>";
  }
  restore_error_handler();
?>
```

STORING ERRORS IN A LOG FILE

WE CAN SEND OUR ERROR MESSAGES TO A .LOG FILE. IN THIS CASE, THE NAME OF THE FILE WOULD NOT BE error.log. BECAUSE IT ALREADY EXISTS AND IS HANDLING APACHE ERRORS.

```
function handlingErrors($eLevel, $eMessage, $eFile, $eLine) {
    error_log("$eMessage in $eFile, line $eLine", // message
        3,                                        // append mode
        "c:/xampp/apache/logs/user_errors");      // file route/name
}
```

IN ADDITION, WE CAN ADD THE USER NAME (`get_current_user()`), THE IP (`$_SERVER['REMOTE_ADDR']`) OF THE CLIENT THAT LAUNCHED THE SCRIPT, THE DATE AND ANY OTHER INFORMATION.

STORING ERRORS IN A LOG FILE

```
function handlingErrors($eLevel, $eMessage, $eFile, $eLine) {

   $newMessage = "Date: ".date("H:i d-m-Y ").$eMessage.
                 " in file ".$eFile." line ".$eLine.
                 " User: ".get_current_user()." from IP: ".
                 $_SERVER['REMOTE_ADDR'];

   error_log("$newMessage in $eFile, line $eLine", 3,

      "c:/xampp/apache/logs/user_errors");
}
```