PHP Introduction
Exception Handling

DWES UD2

1.- Exceptions

An exception in PHP occurs when the application attempts to perform a task and is unable to do so.

The exception will stop execution unless we **catch** and **handle** it.

By capturing and handling an exception, we can:

- Avoid displaying unwanted error messages to the end user.
- Prevent the application from crashing suddenly.

1.- Exceptions

In PHP 5, there is the **Exception** class (it's also available in PHP 7 and PHP8)

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

In PHP 7 and PHP 8 (not available in PHP5), we have the **Throwable** class instead. It covers both exceptions and internal errors.

1.- Exceptions

In case we are not sure whether the server supports PHP5 or PHP7, we can introduce both clauses in our try...catch:

```
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){
    // Only in PHP 5, won't be reached in PHP 7
}
```

1.- Exceptions

Let's see an example. If we run the following code:

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

The output will be like this:

The inverse of 10 is 0.1

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>
```

The output will be like this:

Fatal error: Uncaught DivisionByZeroError: Division by zero

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>
```

The output will be like this:

Fatal error: Uncaught DivisionByZeroError: Division by zero

1.- Exceptions

We can handle the exception using try...catch:

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

WE WILL GET THIS OUTPUT:

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.

1.- Exceptions

Whether you use Throwable or Exception, both offer you some methods that can help you https://www.php.net/manual/en/class.exception.php

```
getMessage () ;    // returns the exception message

getCode ();        // returns the exception code

getFile ();        // get the name of the file in which the exception was created

getLine ();        // Get line number where the exception was created

__toString ();     // Returns the string representation of the exception
```

1.- Exceptions

```php
<?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

1.- Exceptions

We can throw exceptions inside our application

```php
<?php
//create function with an exception
function checkNum($number) {
  if($number>1) {
    throw new Exception("Value must be 1 or below");
  }
  return true;
}

//trigger exception in a "try" block
try {
  checkNum(2);
  //If the exception is thrown, this text will not be shown
  echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

1.- Exceptions

We can create a custom exception class by extending the Exception class.

The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

```php
<?php
class customException extends Exception {
  public function errorMessage() {
    //error message
    $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
    .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
    return $errorMsg;
  }
}

$email = "someone@example...com";

try {
  //check if
  if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE) {
    //throw exception if email is not valid
    throw new customException($email);
  }
}

catch (customException $e) {
  //display custom message
  echo $e->errorMessage();
}
?>
```

2.- Warnings

Some errors can't be handled. For example:

```
<?php
function anverse($number) {
 $anverseNumber = 1 / $number;
 return $anverseumber;    // ERROR
}
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 **5**

The inverse of 10 is

2.- Warnings

- A Warning is a non-fatal error that raises a message but does not stop execution.
- A Warning cannot be handled by a try...catch structure.
- Warnings can be disabled in the php.ini file configuration or by using the error_reporting (E_ERROR) function.
- However, handling them is considered good developer etiquette.

2.- Warnings

We can handle warnings using set_error_handler to define which function will be used to handle both errors and exceptions:

```
set_error_handler("handleErrors");
```

Then, the definition of the function can be something like this:

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

We will need to restore automatic exception handling at the end of the script.

```
restore_error_handler();
```

2.- Warnings

```
<?php
 function handleErrors($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("handleErrors");
 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();
?>
```

3.- Storing errors in the log file

We can send error messages to a log file.

In this case, the file name should NOT be error.log because one already exists and is managed by Apache.

```
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
}
```

Additionally, you can add the username (get_current_user()), the IP ($_SERVER['REMOTE_ADDR']) of the client that launched the script, the date, and other available information.

3.- Storing errors in the log file

```
function handleErrors($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");
}
```

Exceptions tree

We can check the available exceptions using the following script:

https://gist.github.com/mlocati/249f07b074a0de339d4d1ca980848e6a

The output can be checked here:

https://3v4l.org/sDMsv

Exercises

- Create a script containing a function that adds two numbers passed as parameters. Inside the function, check that the parameters received are numbers. If they aren't, throw an exception and handle it in the main program.

- Create a script that includes a class that extends Exception, and modify the message displayed on the screen to your liking. The script must also contain a function that divides two numbers passed as parameters. Inside the function, check that the parameters received are numbers and that the divisor is not zero. If not, throw an exception and handle it in the main program.