Exception Handling in PHP

Exception handling is a powerful mechanism of PHP, which is used to handle runtime errors (runtime errors are called exceptions). So that the normal flow of the application can be maintained.

The main purpose of using exception handling is **to maintain the normal execution of the application**.

What is an Exception?

An exception is an unexcepted outcome/result of a program, which can be handled by the program itself. Basically, an exception disrupts the normal flow of the program. But it is different from an error because an exception can be handled, whereas an error cannot be handled by the program itself.

In other words, - "An unexpected result of a program is an exception, which can be handled by the program itself." Exceptions can be thrown and caught in PHP.

Why needs Exception Handling?

PHP provides a powerful mechanism, exception handling. It allows you to handle runtime errors such as IOException, SQLException, ClassNotFoundException, and more. A most popular example of exception handling is - divide by zero exception, which is an arithmetic exception.

> Note: Exception handling is required when an exception interrupts the normal execution of the program or application.

try -

The try block contains the code that may have an exception or where an exception can arise. When an exception occurs inside the try block during runtime of code, it is caught and resolved in catch block. The try block must be followed by catch or finally block. A try block can be followed by minimum one and maximum any number of catch blocks.

catch -

The catch block contains the code that executes when a specified exception is thrown. It is always used with a try block, not alone. When an exception occurs, PHP finds the matching catch block.

throw -

It is a keyword used to throw an exception. It also helps to list all the exceptions that a function throws but does not handle itself.

Remember that each throw must have at least one "catch".

finally -

It is used in place of a catch block or after a catch block.The finally block contains a code, which is used for clean-up activity in PHP. Basically, it executes the essential code of the program.

What happens when an exception is triggered -

- The current state of code is saved.
- The execution of the code is switched to a predefined exception handler function.
- Depending on the situation, the handler can halt the execution of program, resume the execution from the saved code state, or continue the execution of the code from another location in the code.

Advantage of Exception Handling over Error Handling

1)Grouping of error types

2)Keep error handling and normal code separate -

Examples

Learn with the help of examples how exception handling works in PHP-

Example 1

Let's take an example to explain the common flow of throw and try-catch block:

```
<?php

// PHP Program to illustrate normal
// try catch block code
function demo($var) {

    echo " Before try block";
```

```
    try {
        echo "\n Inside try block";

        // If var is zerothen only if will be executed
        if($var == 0) {

            // If var is zero then only exception is thrown
            throw new Exception('Number is zero.');

            // This line will never be executed
            echo "\n After throw it will never be executed";
        }
    }

    // Catch block will be executed only
    // When Exception has been thrown by try block
    catch(Exception $e) {
        echo "\n Exception Caught" . $e->getMessage();
    }
    finally {
        echo "\n Here cleanup activity will be done";
    }

    // This line will be executed whether
    // Exception has been thrown or not
    echo "\n After catch it will be always executed";
}

// Exception will not be raised
demo(5);

// Exception will be raised here
demo(0);
?>
```

Output:

```
 Before try block
 Inside try block
 Here cleanup activity will be done
 After catch (will be always executed)
 Before try block
 Inside try block
 Exception CaughtNumber is zero.
 Here cleanup activity will be done
 After catch (will be always executed)
```

Example 2: Creating Custom Exception

You can create user-defined exception by extending Exception class. Look at the code below to learn how create user-defined exception -

```
1.  <?php
2.  class DivideByZeroException extends Exception { }
3.  class DivideByNegativeNoException extends Exception { }
4.  function checkdivisor($dividend, $divisor){
5.     // Throw exception if divisor is zero
6.    try {
7.       if ($divisor == 0) {
8.         throw new DivideByZeroException;
9.       }
10.      else if ($divisor < 0) {
11.        throw new DivideByNegativeNoException;
12.      }
13.      else {
14.        $result = $dividend / $divisor;
15.        echo "Result of division = $result </br>";
16.      }
17.   }
18.   catch (DivideByZeroException $dze) {
19.       echo "Divide by Zero Exception! </br>";
20.   }
21.   catch (DivideByNegativeNoException $dnne) {
22.       echo "Divide by Negative Number Exception </br>";
23.   }
24.   catch (Exception $ex) {
25.       echo "Unknown Exception";
26.   }
27. }
28.
29.   checkdivisor(18, 3);
30.   checkdivisor(34, -6);
31.   checkdivisor(27, 0);
32. ?>
```

Output:

```
Result of division = 6
Divide by Zero Exception!
Divide by Negative Number Exception!
```

- Using Custom Exception Class

php

```php
<?php
class myException extends Exception {
    function get_Message() {

        // Error message
        $errorMsg = 'Error on line '.$this->getLine().
                    ' in '.$this->getFile()
        .$this->getMessage().' is number zero';
        return $errorMsg;
    }
}

function demo($a) {
    try {

        // Check if
        if($a == 0) {
            throw new myException($a);
        }
    }

    catch (myException $e) {

        // Display custom message
        echo $e->get_Message();
    }
}

// This will not generate any exception
demo(5);

// It will cause an exception
demo(0);
?>
```

Output:

```
Error on line 20 in /home/45ae8dc582d50df2790517e912980806.php0 is
number zero
```

- **Set a Top Level Exception Handler:** The set_exception_handler() function set all user-defined functions to all uncaught exceptions.

php

```php
<?php

// PHP Program to illustrate normal
// try catch block code

// Function for Uncaught Exception
function myException($exception) {

    // Details of Uncaught Exception
    echo "\nException: " . $exception->getMessage();
}

// Set Uncaught Exception handler
set_exception_handler('myException');
function demo($var) {

    echo " Before try block";
    try {
        echo "\n Inside try block";

        // If var is zero then only if will be executed
        if($var == 0)
        {

            // If var is zero then only exception is thrown
            throw new Exception('Number is zero.');

            // This line will never be executed
            echo "\n After throw (it will never be
executed)";
        }
    }

    // Catch block will be executed only
```

```
    // When Exception has been thrown by try block
    catch(Exception $e) {
        echo "\n Exception Caught", $e->getMessage();
    }

    // This line will be executed whether
    // Exception has been thrown or not
    echo "\n After catch (will be always executed)";

    if($var < 0) {

        // Uncaught Exception
        throw new Exception('Uncaught Exception occurred');
    }
}

// Exception will not be raised
demo(5);

// Exception will be raised here
demo(0);

// Uncaught Exception
demo (-3);
?>
```

Output:

```
Before try block

Inside try block

After catch (will be always executed)

Before try block

Inside try block

Exception CaughtNumber is zero.

After catch (will be always executed)

Before try block

Inside try block

After catch (will be always executed)

Exception: Uncaught Exception occurred
```