PHP – Exceptions

Prior to version 7, PHP parser used to report errors in response to various conditions. Each error used to be of a certain predefined type. PHP7 has changed the mechanism of error reporting. Instead of traditional error reporting, most errors are now reported by throwing error exceptions.

The exception handling mechanism in PHP is similar to many other languages, and is implemented with the **try, catch, throw** and **finally** keywords.

The Throwable Interface

Exceptions in PHP implements the **Throwable interface**. The Throwable interface acts as the base for any object that can be thrown via throw statement, including Error and Exception objects.

A user defined class cannot implement Throwable interface directly. Instead, to declare a user defined exception class, it must extend the **Exception class**.

PHP code with potential exceptions is surrounded in a **try** block. An exception object is thrown if it is found, to facilitate the catching of potential exceptions. Each **try** must have at least one corresponding **catch** or **finally** block. Moreover, there may be more than one catch/finally blocks corresponding to a **try** block.

```
try {

   // throw errors in the try-block
   // if an error occurs we can throw an exception
   throw new Exception('this is an error.');
}
catch(Exception $e) {

   // catch the throws in the catch-block
   // do something with the exception object, eg.
   // display its message
   echo 'Error message: ' .$e->getMessage();
}
```

If an exception is thrown and there is no **catch** block, the exception will "bubble up" until it finds a matching **catch** block. If the call stack is unwound all the way to the global scope without encountering a matching catch block, a global exception handler will be called (if it is set) otherwise the program will terminate with a fatal error.

set_exception_handler

This function sets the default exception handler if an exception is not caught within a try/catch block. After the callback is executed, the program execution will stop.

```
set_exception_handler(?callable $callback): ?callable
```

The $callback parameter is the name of the function to be called when an uncaught exception occurs. This function must be defined before calling set_exception_handler(). This handler function needs to accept one parameter, which will be the exception object that was thrown.

The function returns the name of the previously defined exception handler, or NULL on error. If no previous handler was defined, NULL is also returned.

Example

Take a look at the following example –

</> Open Compiler

```
<?php
   function handler($ex) {
      echo "Uncaught exception is : " , $ex->getMessage(), "\n";
   }

   set_exception_handler('handler');
   throw new Exception('Not Found Exception');
   echo "not included Executed\n";
?>
```

It will produce the following **output** –

```
Uncaught exception is : Not Found Exception
```


SPL Exceptions

Standard PHP library contains predefined exceptions –

Sr.No	Predefined Exceptions
1	**LogicException** Exception that represents error in the program logic.
2	**BadFunctionCallException** Exception thrown if a callback refers to an undefined function or if some arguments are missing.
3	**BadMethodCallException** Exception thrown if a callback refers to an undefined method or if some arguments are missing.
4	**DomainException** Exception thrown if a value does not adhere to a defined valid data domain.
5	**InvalidArgumentException** Exception thrown if an argument is not of the expected type.
6	**LengthException** Exception thrown if a length is invalid.
7	**OutOfRangeException** Exception thrown when an illegal index was requested.
8	**RuntimeException** Exception thrown if an error which can only be found on runtime occurs.
9	**OutOfBoundsException** Exception thrown if a value is not a valid key.
10	**OverflowException** Exception thrown when adding an element to a full container.
11	**RangeException** Exception thrown to indicate range errors during program execution. An arithmetic error other than under/overflow.
12	**UnderflowException** Exception thrown when performing an invalid operation on an empty container, such as removing an element.
13	**UnexpectedValueException** Exception thrown if a value does not match with a set of values.

User-defined Exception

You can define a custom exception class that extends the base Exception class. Following script defines a custom exception class called myException. This type of exception is thrown if value of $num is less than 0 or greater than 100.

Example

The getMessage() method of Exception class returns the error message and getLine() method returns line of code in which exception appears.

</> Open Compiler

```
<?php
   class myException extends Exception {
      function message() {
         return "error : ". $this->getMessage(). "in line no". $this->getLine();
      }
   }
   $num=125;
   try {
      if ($num>100 || $num<0)
      throw new myException("$num is invalid number");
      else
      echo "$num is a valid number";
   }
   catch (myException $m) {
      echo $m->message();
   }
?>
```

Run the above code with **$num=125** and **$num=90** to get an error message and a message of valid number –

```
error : 125 is invalid number in line no 10
```