# 例外処理

## フォームを作る

まず、以下のフォームを作って、エラーの起こる様子を見ます。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>PHP</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<h1></h1>

</header>

<article>

<form action="reigai2.php" method="post">

<p>数字を入力してください</p>

<p>

<input type="text" name="left"> ÷

<input type="text" name="right"> ＝

<input type="submit" value="答え">

</p>

</form>

</article>

<hr>

<footer>

<small>&copy; 2017 Seiichi Nukayama</small>

</footer>

</body>

</html>

これを受け取るのは、以下のphpスクリプトです。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>PHP</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<h1></h1>

</header>

<article>

<?php

function divnum($a, $b){

$c = $a / $b;

return $c;

}

$x = ($\_POST['left']);

$y = ($\_POST['right']);

$z = divnum($x, $y);

print "$x ÷ $y = $z \n";

?>

<p><a href="reigai1.html">もどる</a></p>

</article>

<hr>

<footer>

<small>&copy; 2017 Seiichi Nukayama</small>

</footer>

</body>

</html>

4 ÷ 2 なら 2 と表示されますが、4 ÷ 0 なら、INF（無限大） と表示されます。

XAMPPコントロールパネルのapacheの行のlog項目をクリックし、apache2のエラーログ(error.log)を見ると以下のような内容になっています。

PHP Warning: Division by zero in /home/se-ichi/public\_html/phpbook/newdoc/reigai2.php on line 17

ゼロで割っているという警告がでていることがわかります。

## エラーを例外に変える

phpスクリプト内で起こったエラーに応じた処理を作るには、まず、エラーを例外としてスクリプト自身に通知します。

処理の最初に以下のように記述します。

**set\_error\_handler("myErrorSyori");**

$x = ($\_POST['left']);

$y = ($\_POST['right']);

（以下、略）

Set\_error\_handler関数は、エラーが起こったときに引数に指定した関数に処理を渡します。

つまり自分の好きなように関数をつくって、エラーが起こったときの処理を記述できるわけです。

次に、エラーのときの処理を myErrorSyori 関数に記述します。

function myErrorSyori($errcode, $msg, $file, $line) {

throw new ErrorException($msg, 0, $errcode, $file, $line);

}

この関数のはたらきは、エラーを例外に変換することです。

ErrorExceptionというクラスが例外を作成しています。ここでは、エラーの内容も表示しようとして、$errcode や $msg も引数にしています。そういうのが必要なければ、以下のような内容でもいいです。

function myErrorSyori () {

Throw new ErrorException(‘無理っス！’);

}

その結果できた例外オブジェクトをスロー（throw）しています。このスローされた例外オブジェクトを次のtry…catch…で捕捉します。

(注）「phpの絵本」p154では、以下のようになってます。

Function myErrorProc($errcode, $msg, $file, $line) {

// 出力する種類のエラーでないときは、スローしない。

If ( ! (error\_reporting() & $errcode ) {

Return;

}

Throw new ErrorException($msg, 0, $errcoce, $file, $line);

}

この **If ( ! (error\_reporting() & $errcode )** についてですが、

error\_reporting() …… 現在のエラー設定が値として取り出せます。

E\_ALL & ~E\_NOTICE（すべてのエラーを出力するが、NOTIEは除く）になっているはずです。また、

error\_reporting(E\_ALL & ~E\_NOTICE & ~E\_DEPRICATED) ;

などと、エラー設定を変更することも可能です。

$errcode は、このスクリプトの実行時に出たエラーコードです。

このふたつを「&」（論理和）によってつなげると、両方が成立した場合という意味になります。それの先頭に「!」がついているので、ここでは、「両方が成立しなかったら」ということになり、片方のエラーが出ても、もう片方が成立していなかったら、エラーとはしないということになります。

## エラーの捕捉

エラーが発生しそうなところに、try…catch…を記述します。

function divnum($a, $b){

try {

$c = $a / $b; ------ (1)

} catch (Exception $e) { ------ (2)

$c = "計算できません。（" . $e->getMessage() . "）"; ------ (3)

}

return $c;

}

1. でエラーが発生すると、saet\_error\_handlerで指定しておいた関数myErrorSyoriが働き、発生したエラーを例外オブジェクトに変換し、それをこのphpスクリプト内に投げます。
2. 投げられた例外オブジェクトは、このcatchで捕捉され、その内容が$eという変数に格納されます。
3. $eは例外オブジェクトです。それにおさめられたメッセージは$e->getMessage()で取り出すことができます。

完成したreigai2.php を以下に示します。

<!DOCTYPE html>

<html lang="ja">

<head>

<meta charset="utf-8">

<title>PHP</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<header>

<h1></h1>

</header>

<article>

<?php

function divnum($a, $b){

try {

$c = $a / $b;

} catch (Exception $e) {

$c = "計算できません。{$e->getMessage()} <br>\n";

}

return $c;

}

function mySyori() {

throw new ErrorException('無理っス!');

}

function myErrorSyori($errcode, $msg, $file, $line) {

throw new ErrorException($msg, 0, $errcode, $file, $line);

}

// set\_error\_handler("myErrorSyori");

set\_error\_handler("mySyori");

$x = ($\_POST['left']);

$y = ($\_POST['right']);

$z = divnum($x, $y);

print "$x ÷ $y = $z <br> \n";

?>

<p><a href="reigai1.html">もどる</a></p>

</article>

<hr>

<footer>

<small>&copy; 2017 Seiichi Nukayama</small>

</footer>

</body>

</html>

途中、set\_error\_handlerで２種類の処理を指定できるようにしてあります。引数の関数名を書き換えるだけです。