
Bài 13

Xử lý ngoại lệ

Module: **ADVANCED PROGRAMMING WITH PHP 2.0**

Mục tiêu

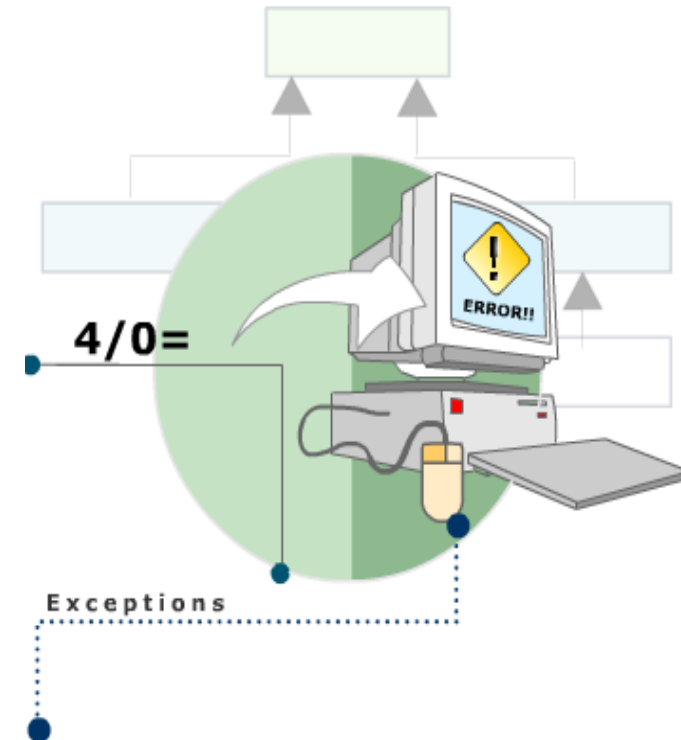


- Trình bày được ý nghĩa của việc xử lý ngoại lệ
- Sử dụng được try-catch để bắt ngoại lệ
- Tung được ngoại lệ phù hợp
- Sử dụng được các lớp ngoại lệ thông dụng có sẵn
- Định nghĩa được các lớp ngoại lệ tùy biến

Ngoại lệ



- Ngoại lệ là các lỗi phát sinh trong quá trình thực thi
- Ví dụ:
 - Lỗi khi lái xe
 - Lỗi khi máy tính xử lý phép chia cho 0





Các kiểu lỗi thường xảy ra

- Lỗi cú pháp
- Lỗi khi chạy chương trình
- Lỗi về tính logic cấu trúc của chương trình

Lỗi cú pháp



- Lỗi cú pháp (Syntax Error) xảy ra tại thời gian biên dịch trong các ngôn ngữ chương trình truyền thống và tại thời gian phiên dịch trong Javascript.
- Ví dụ, dòng sau gây ra một lỗi cú pháp bởi vì nó thiếu dấu ngoặc đơn đóng:

```
<?php
function isNum($number) {
    if($number>1
        return true;
}

isNum(2);
?>
```

Lỗi khi chạy chương trình



- Lỗi trong khi chạy chương trình (Runtime Error) xảy ra trong suốt thời gian thực thi
- Ví dụ, dòng sau tạo một Runtime Error bởi vì ở đây cú pháp là đúng, nhưng trong khi chạy, nó cố gắng gọi một phương thức mà không tồn tại.

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

/* trigger exception */
checkNumber(2);
?>
```

Lỗi logic



- Lỗi về tính logic của cấu trúc chương trình (Logic Error) là kiểu lỗi khó để có thể tìm dấu vết.
- Xảy ra khi bạn tạo một lỗi về tính logic mà điều khiển script của bạn và không nhận được kết quả như mong đợi.
- Khó nắm bắt được các lỗi này, bởi vì nó phụ thuộc vào yêu cầu và kiểu logic mà bạn đặt vào chương trình.



Xử lý ngoại lệ trong PHP

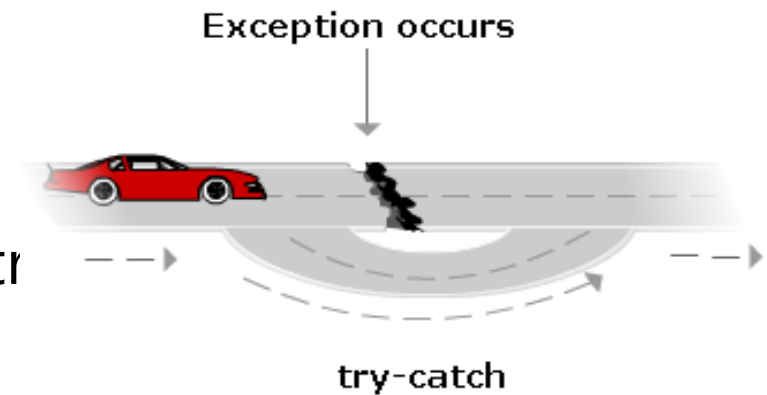
- Khả năng bắt và giải quyết ngoại lệ sử dụng catch.
- Khả năng tách logic xử lý ngoại lệ trong một hàm ra khỏi phần còn lại của hàm sử dụng try.
- Khả năng tạo và ném ngoại lệ sử dụng throw.

Sử dụng khối try-catch

- Khối try-catch dùng để
 - Tách phần giải quyết lỗi ra khỏi phần có thể sinh lỗi
 - Quy định các loại ngoại lệ được bắt tại mức thực thi hiện hành
- Cú pháp

```
try {  
    Statement_1;  
    Statement_2;  
}  
catch (ExceptionType ObjectName) {  
    Statement_1;  
}
```

- Mã liên quan đến thuật toán nằm trong khối tr
- Mã giải quyết lỗi đặt trong các khối catch



Demo



```
<?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 */
checkNum(2);
?>
```

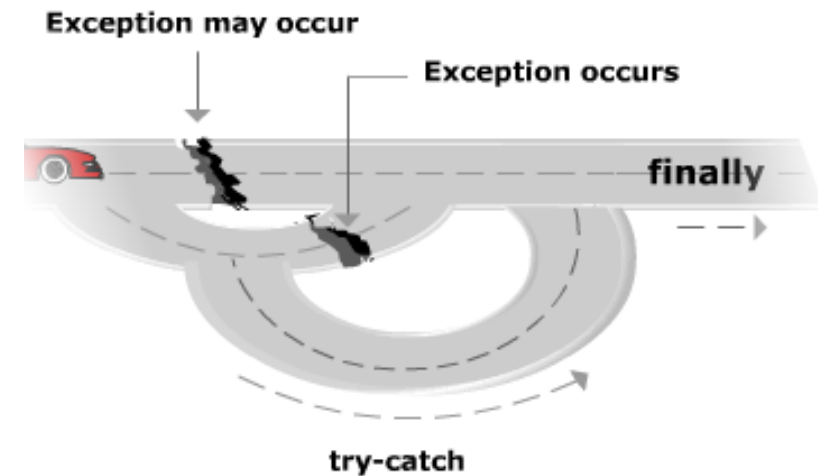
Fatal error: Uncaught exception 'Exception'
with message 'Value must be 1 or below' in C:\webfolder\test.php:6
Stack trace: #0 C:\webfolder\test.php(12):
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6

Khối **finally**



- Khối **finally** mà sẽ luôn luôn thực thi vô điều kiện sau **try/catch**.

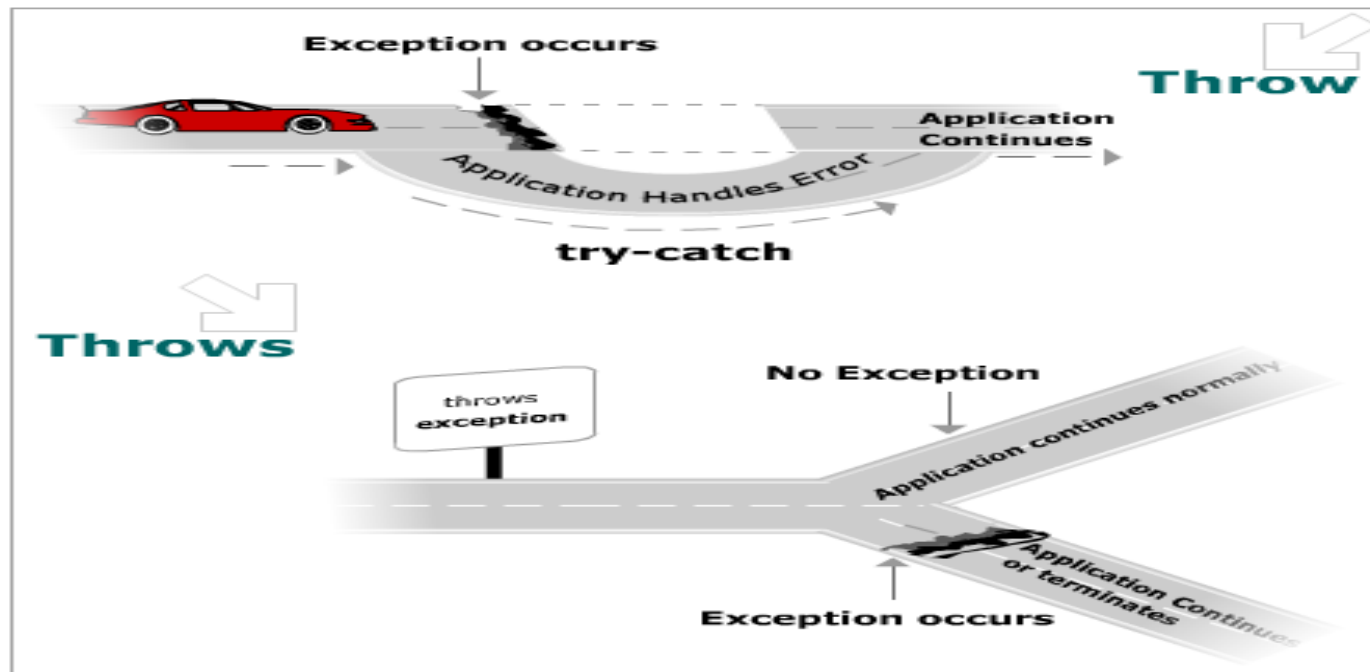
```
try {  
    Statement_1;  
    Statement_2;  
}  
catch (ExceptionType ObjectName) {  
    Statement_1;  
}  
finally {  
    //Clean up code  
    Statement_1;  
}
```



Ném ngoại lệ



- Sử dụng lệnh **throw** để ném các lỗi Runtime Error có sẵn hoặc các lỗi do người dùng tự định nghĩa. Sau đó các lỗi này có thể được bắt và bạn có thể thực hiện một hành động hợp lý.



Ném ngoại lệ



- Để ném một ngoại lệ, ta dùng từ khoá throw kèm theo đối tượng mà ta định ném
- Ta có thể dùng mọi thứ làm ngoại lệ, kể cả giá trị thuộc kiểu có sẵn

```
<?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();
}
?>
```

Ngoại lệ do người dùng tự tạo

- Để tạo một ngoại lệ của người dùng, bạn phải tạo một lớp là mở rộng của lớp Exception có sẵn.
- Lớp ngoại lệ do người dùng tự định nghĩa kế thừa từ lớp exception và chúng ta có thể thêm hàm xử lý.

```
<?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();
}
?>
```



Tóm tắt bài học

- Ngoại lệ là các lỗi phát sinh trong quá trình thực thi
- Xử lý ngoại lệ với khối try-catch-finally
- Sử dụng lệnh throw để ném các lỗi

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Chuỗi vs Regex***