

PHP Cơ bản

- Include file trong PHP
- Cookie trong PHP
- Session trong PHP

- Include tập tin phía server thường được dùng để tạo các hàm (function), các tiêu đề (header), footer, hoặc các phần tử một lần, sau đó được tái sử dụng trong nhiều trang khác nhau.
- Bạn có thể chèn nội dung của một file vào file PHP trước server thực thi nó, bằng việc dùng hàm `include()` hoặc hàm `require()`.
- Hai hàm trên rất giống nhau, chỉ khác nhau về cách xử lý lỗi:
 - Hàm `include()` sinh ra một cảnh báo (Warning), nhưng script vẫn tiếp tục được thực thi.
 - Hàm `require()` sinh ra một Fatal Error, và script sẽ dừng lại ở đó.

- Ví dụ dùng hàm `include()`
 - Giả sử bạn có một file header có tên là "header.php". Để include header file trong một trang, hãy sử dụng hàm `include()` như sau:

```
<html>
    <body>
        <?php
            include ("header.php");
        ?>
        <h1>Welcome to my home page</h1>
        <p>Some text</p>
    </body>
</html>
```

- Ví dụ dùng hàm include()
 - Code của hàm include():

```
<html>
  <body>
    <?php
      include("wrongFile.php");
      echo "Hello World!";
      include("File2.php");
    ?>
  </body>
</html>
```

- **Warning:** include(wrongFile.php) [function.include]: failed to open stream: No such file or directory in C:\home\website\test.php on line 5
- **Warning:** include() [function.include]: Failed opening 'wrongFile.php' for inclusion (include_path='.;C:\php5\pear') in C:\home\website\test.php on line 5Hello World!

- Ví dụ dùng hàm require()

```
<html>
    <body>
        <?php
            require("wrongFile.php");
            echo "Hello World!";
        ?>
    </body>
</html>
```

```
include("wrong.php");
echo "Hello";
```

Warning: require(wrong.php) [[function.require](#)]: failed to open stream: No such file or directory in **C:\wamp\www\test.php** on line 3

```
require("wrong.php");
echo "Hello";
```

Fatal error: require() [[function.require](#)]: Failed opening required 'wrong.php' (include_path='.;C:\php5\pear') in **C:\wamp\www\test.php** on line 3

- Cookie là một file nhỏ(tối đa 4KB) được server sử dụng để lưu các thông tin tại phía Trình duyệt.
- Cookie thường được dùng để lưu trữ thông tin về người sử dụng khi họ truy cập vào website.
- Mỗi lần máy tính gửi yêu cầu một trang web từ trình duyệt, nó sẽ gửi kèm theo cả cookie.
- Khả năng tạo cookie phụ thuộc vào trình duyệt và sự cho phép của người sử dụng.

- Hàm setcookie() được dùng để tạo một cookie.
 - Cú pháp:

```
setcookie(name, value, expire, [path], [domain]);
```

- Ví dụ tạo một cookie:

```
<?php
    setcookie('user', 'Alex Porter', time() + 60 * 60);
    //Thời hạn sử dụng của user là một giờ kể từ lệnh này
?>
<html>
    <body> </body>
</html>
```

- **Chú ý:** Hàm setcookie() phải xuất hiện trước thẻ <html>, trước lệnh echo, print.

- Cách lấy giá trị trong Cookie
 - Sử dụng biến **\$_COOKIE** để lấy giá trị trong cookie do trình duyệt gửi đến.
 - Bất kỳ khi nào có cookie gửi từ trình duyệt, Server sẽ tự động lưu cookie vào mảng global **\$_COOKIE**.
 - Trong ví dụ dưới đây, chúng ta đã lấy ra giá trị của biến cookie có tên là "user" và hiển thị nó lên trang web:

```
<?php
    //in ra giá trị của một cookie
    echo $_COOKIE["user"];
    //in ra giá trị của tất cả các cookie
    print_r($_COOKIE);
?>
```

- Làm thế nào để xóa Coookie?
 - Khi muốn xóa Cookie thì chỉ cần thiết lập expiration(thời hạn) lùi về quá khứ .

```
<?php  
    // Thiết lập cookie đã hết hạn từ 1 giờ cách đây  
    setcookie ("user", "", time () -3600);  
?>
```

- Cứ mỗi người dùng khi làm việc với website đều được web server tạo ra một phiên làm việc riêng của người đó.
- Session là một cách lưu trữ thông tin trên web server để có thể chia sẻ chúng cho tất cả các trang trong cùng ứng dụng.
- Thông tin lưu trữ trong biến **`$_SESSION`** là tạm thời, nó sẽ bị hủy khi người truy cập không còn làm việc với website (đóng cửa sổ trình duyệt, truy cập một website khác, hoặc timeout).
- Session được dùng để quản lý thông tin về mỗi người dùng người dùng riêng biệt.

- Khởi động session
 - Trước khi sử dụng session của PHP để lưu trữ thông tin, cần phải khởi động session bằng hàm dựng sẵn `session_start()`.

```
<?php
    session_start();
?>
<html>
    <body> </body>
</html>
```

Đoạn code trên sẽ đăng ký session của người dùng với server, cho phép bạn bắt đầu quá trình lưu trữ thông tin người dùng, và gán một UID cho session đó của người dùng.

- **Chú ý:** Hàm `session_start()` function phải xuất hiện trước thẻ `<html>`:

- Lưu trữ và lấy dữ liệu trong biến Session
 - Sử dụng biến **\$_SESSION**

```
<?php
    session_start();
    //Lưu trữ dữ liệu vào session
    $_SESSION['views']=1;
?>
<html> <body>
<?php
    //Lấy dữ liệu trong session
    echo "Pageviews=". $_SESSION['views'];
?>
</body> </html>
```

Kết quả:
Pageviews=1

- Nếu bạn muốn xóa một vài dữ liệu trong session, bạn có thể sử dụng hàm unset() hoặc hàm session_destroy().
- Hàm unset() function được dùng để giải phóng một biến session theo tên:

```
<?php unset($_SESSION['views']); ?>
```

- Bạn cũng có thể hủy toàn bộ session bằng cách gọi hàm session_destroy():

```
<?php session_destroy(); ?>
```

- **Chú ý:** Hàm session_destroy() sẽ reset session của bạn và bạn sẽ mất tất cả dữ liệu đã lưu trong session.

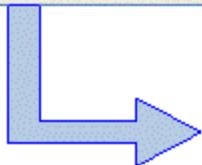
- Quản lý lỗi mặc định trong PHP
 - Khi có lỗi xảy ra, một dòng thông báo lỗi gồm các thông tin: tên file, vị trí dòng phát sinh lỗi, ... được gửi cho phía trình duyệt.

```
Parse error: parse error in C:\wamp\www\test.php on line 4
```

- Chứa nhiều rủi ro về bảo mật.
- Có 3 cách để xử lý lỗi
 - Sử dụng hàm **die()**
 - Tạo hàm xử lý lỗi riêng & Error Trigger
 - Error logging / reporting

- Sử dụng hàm die()

```
<?php  
    $file = fopen("Wrong.txt", "r");  
?>
```



Warning: fopen(Wrong.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\wamp\www\test.php on line 2

```
if(!file_exists("Wrong.txt")) {  
    die("File not found");  
} else {  
    $file = fopen("Wrong.txt", "r");  
}
```



File not found

- **Tạo hàm xử lý lỗi riêng**
 - Là hàm đặc biệt, được gọi khi có lỗi xảy ra
 - Phải xử lý tối thiểu hai tham số: `error_level` và `error_message`
- `error_function_name(error_level , error_message,
error_file, error_line, error_context);`

Tham số	Mô tả
<code>error_level</code>	Mã lỗi, là tham số bắt buộc. (xem bảng các mã lỗi)
<code>error_message</code>	Nội dung thông báo, là tham số bắt buộc.
<code>error_file</code>	Tên file xảy ra lỗi. Tham số tùy chọn
<code>error_line</code>	Vị trí của dòng xảy ra lỗi. Tham số tùy chọn
<code>error_context</code>	Mảng chứa danh sách biến và giá trị đang được sử dụng khi lỗi xảy ra

- Bảng mã lỗi

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. The script is not halted
8	E_NOTICE	Run-time notices. something that might be an error.
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by trigger_error();
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle
8191	E_ALL	All errors and warnings, except level E_STRICT

- Thiết lập trình quản lý lỗi (Set error handler)
 - Mặc định trình quản lý lỗi cho PHP là trình quản lý lỗi dựng sẵn (built-in)
 - Để thay đổi trình quản lý lỗi: `set_error_handler("...")`

```
function customError($errNo, $errText) {  
    echo "<b>Error:</b> [$errNo] $errText <br/>";  
}  
  
//set error handler  
set_error_handler("customError");  
  
//Trying to output variable that does not exist:  
echo ($test);
```

Error: [8] Undefined variable: test

- Gây ra lỗi

- Có thể làm phát sinh ra lỗi khi người dùng nhập thông tin sai, bằng việc sử dụng hàm `trigger_error()`
- Hàm rất có ích khi bạn cần đối phó lại các ngoại lệ cụ thể lúc thực thi

```
//Ex: error occurs if the test > 1.  
$test=2;  
if ($test>1) {  
    trigger_error("Value must be 1 or below");  
}
```



Notice: Value must be 1 or below
in C:\wamp\www\test.php on line 6

- Sử dụng **error types** trong tham số thứ 2:
 - **E_USER_ERROR** - Fatal user-generated run-time **error**. Errors that can not be recovered from. Execution of the script is halted
 - **E_USER_WARNING** - Non-fatal user-generated run-time **warning**. Execution of the script is not halted
 - **E_USER_NOTICE** - Default. User-generated run-time **notice**. The script found something that might be an error.

- Ví dụ:

```
function customError($errno, $errstr) {  
    echo "<b>Error:</b> [$errno] $errstr<br />";  
    echo "Ending Script";  
    die();  
}  
  
//set error handler  
set_error_handler("customError", E_USER_WARNING);  
  
//Trigger error  
$test=2;  
if ($test>1) {  
    trigger_error("Value must be 1 or below", E_USER_WARNING);  
}
```



Error: [512] Value must be 1 or below
Ending Script

- Mặc định PHP lưu thông báo lỗi vào một tập tin trên server, hoặc có thể gửi ra một tập tin bên ngoài qua email,... bằng cách sử dụng hàm `error_log()`.
- Cấu hình log trong file `php.ini`
- Hiển thị thông báo lỗi
 - Thêm vào các dòng code sau
 - Hoặc sửa file `php.ini`
 - Để tắt thông báo lỗi cho một tài liệu, thêm vào dòng sau

```
ini_set('display_errors', 1);  
error_reporting(E_ALL);
```

```
error_reporting = E_ALL
```

```
error_reporting(0);
```

- Trong PHP 5 cũng có mô hình ngoại lệ tương tự như các ngôn ngữ lập trình khác.
- Trong PHP, một ngoại lệ có thể được tung ra (thrown) và được bắt lại (caught).
- Ngoại lệ có thể được xử lý
 - Sử dụng try ... catch...
 - Tạo ra trình quản lý ngoại lệ riêng
 - Tung ra và bắt các ngoại lệ
 -

- Cú pháp

```
try
{
    //các đoạn mã có khả năng gây lỗi đặt ở đây
}
catch (kiểu_ngoại_lệ_1 $tên_đối_tượng_1)
{
    //Các câu lệnh xử lý nếu có ngoại lệ xảy ra đặt ở đây
}
catch (kiểu_ngoại_lệ_n $tên_đối_tượng_n)
{
    //Các câu lệnh xử lý nếu có ngoại lệ xảy ra đặt ở đây
}
```

- Mỗi khối *try* phải có ít nhất một khối *catch*.
- Có thể dùng nhiều khối *catch* để bắt nhiều loại ngoại lệ khác nhau.
- Khi có một ngoại lệ xảy ra, PHP sẽ tìm khối *catch* phù hợp cho việc xử lý ngoại lệ đó, và những câu lệnh ở phía sau sẽ không được thực thi.
- Nếu ngoại lệ không được bắt, lỗi Fatal Error được phát sinh kèm theo thông báo ‘*Uncaught Exception*’.
- *throw* được sử dụng để tạo ra một đối tượng ngoại lệ bên trong một hàm. Đối tượng ngoại lệ này được ném ra ngoài hàm đến nơi gọi hàm để xử lý.
- Mỗi *throw* phải có tối thiểu một *catch*.

Sử dụng try, catch, throw

- Ví dụ:

```
function checkNum($number) {  
    if($number>1) {  
        throw new Exception("Value must be 1 or below");  
    }  
    return true;  
}
```

```
try {  
    checkNum(2);  
    echo 'If you see this, the number is 1 or below';  
} catch(Exception $e) {  
    echo 'Message: ' . $e->getMessage();  
}
```

➔ Value must be 1 or below

- Có thể định nghĩa thêm ngoại lệ khi các ngoại lệ có sẵn không đáp ứng được một số tình huống đặc biệt của ứng dụng.
- Các lớp ngoại lệ người dùng dùng định nghĩa thừa kế lớp có sẵn là Exception.
- Cách các ngoại lệ người dùng định nghĩa quản lý và ném ra các lỗi vẫn giống với cách của các ngoại lệ có sẵn.

User Exception

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

```
$email = "someone@inet.vn";  
try {  
    if(filter_var($email, FILTER_VALIDATE_EMAIL) == FALSE) {  
        throw new EmailInvalidException ($email);  
    }  
} catch (EmailInvalidException $e) {  
    echo $e->errorMessage();  
}
```

- Có thể ném ra nhiều ngoại lệ bằng khối if ... else

```
if(điều_kiệu_1) {
    throw new kiểu_ngoại_lệ_1();
} else if (điều_kiệu_2) {
    throw new kiểu_ngoại_lệ_n();
} else if (điều_kiệu_n) {
    throw new kiểu_ngoại_lệ_n();
} else{
    throw new Exception();
}
```

- **Bắt và xử lý nhiều ngoại lệ**

```
try
{
    //các đoạn mã có khả năng gây lỗi đặt ở đây
}
catch (kiểu_ngoại_lệ_1 $tên_đối_tượng_1)
{
    //Các câu lệnh xử lý nếu có ngoại lệ xảy ra đặt ở đây
}
catch (kiểu_ngoại_lệ_n $tên_đối_tượng_n)
{
    //Các câu lệnh xử lý nếu có ngoại lệ xảy ra đặt ở đây
}
Catch (Exception $tên_đối_tượng_n)
{
    //Bắt tất cả các ngoại lệ
}
```