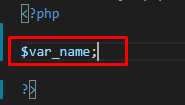
# Biến – Hằng

Định nghĩa: Biến được sử dụng để lưu trữ thông tin trong quá trình xử lý. Có hai loại biến là biến toàn cục và biến cục bộ.

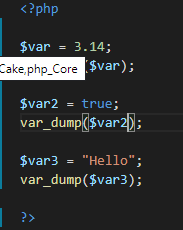
Biến toàn cục nằm ngoài các các hàm và hàm nào cũng có thể sự dụng được biến đó

Biến cục bộ là biến nằm trong hàm và chỉ có hàm đó mới có thể sử dụng biến đó

## Khai báo biến



* Chúng ta có thể gán giá trị cho biến
* Hàm var\_dump() sẽ in ra thông tin của biến và giá trị của biến đó



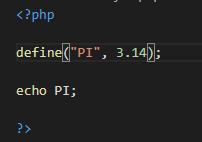


## Hằng

- Hằng cũng được sử dụng để lưu trữ thông tin nhưng giá trị đó không thay đổi.

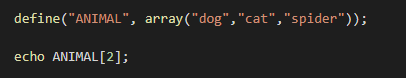
- Nên viết hoa tên hằng số

- Cú pháp: 





- Từ PHP 7 ta có thể khai báo hằng với dạng mảng.





* Ngoài ra ta còn có cách khai báo hằng số khác là: const VAR = 'FOO';

# Echo, print, die, exit, eval

## Echo

- Là câu lệnh dùng để in giá trị ra màn hình. Hay được sử dụng.

## Print

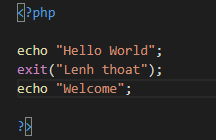
- Là câu lệnh dùng để in giá trị ra màn hình. Ít được sử dụng hơn echo

## Die

- Là câu lệnh dừng hệ thống PHP lại và không dịch tiếp nữa đồng thời cho ra thông báo.

## Exit

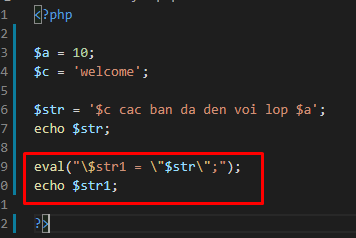
- Giống câu lệnh die.





## Eval

- Chuyển chuỗi sang mã PHP và thực thi nó.





- Với hàm eval nó sẽ chuyển chuỗi $str1 = “$c cac ban da den voi lop $a”; thành mã PHP và khi ta echo $str1 thì máy sẽ chạy string bên trong nó.

- Vì chuỗi $str có **‘’** nên nó sẽ không nhận ra biến lồng trong đó. Nhưng chuỗi $str1 sẽ chuyển đổi sang dạng **“”** nên nó sẽ nhận ra biến lồng bên trong chuỗi.

# Toán tử

## Toán tử gán

- Dấu **=** là toán tử gán. Nó lấy giá trị của bên phải để gán cho biến bên trái.

## Toán tử nối

- Dấu **.** là toán tử nối. Nó nối 2 biến lại với nhau

## Toán tử số học

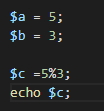
- Dấu **+** là biểu hiện phép cộng

- Dấu **–** là biểu hiện phép trừ

- Dấu **\*** là biểu hiện phép nhân

- Dấu **/** là biểu hiện phép chia

- Dấu **%** là biểu hiện phép chia lấy dư. Tứ là nó sẽ lấy phần dư của phép chia. VD: 5/3=1 dư 2 thì phép chia % sẽ cho ra giá trị là 2

## Toán tử so sánh

- Dấu **>** biểu hiện lớn hơn. VD: $a>$b là $a lớn hơn $b

- Dấu **<** biểu hiện nhỏ hơn. VD: $a<$b là $a nhỏ hơn $b

- Dấu **>=** biểu hiện lớn hơn. VD: $a>=$b là $a lớn hơn hoặc bằng $b

- Dấu **<=** biểu hiện lớn hơn. VD: $a<=$b là $a nhỏ hơn hoặc bằng $b

- Dấu **==** biểu hiện lớn hơn. VD: $a==$b là $a bằng $b

- Dấu **!=** biểu hiện lớn hơn. VD: $a!=$b là $a khác $b

## Toán tử logic

- Dấu **&&** là toán tử và.

- Dấu **||** là toán tử hoặc.

- Dấu **!** là toán tử phủ định.

## Toán tử kết hợp

- Dấu **++** là toán tử tăng. VD: $a=10; $a++; => $a=11

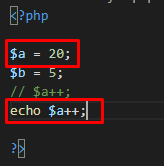
- Dấu **- -** là toán tử giảm. VD: $a=10; $a--; => $a =9

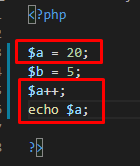
- Dấu **+=, -=, \*=, /=, %=** là toán tử kết hợp. VD: $a = 10; $a += 5; => $a = $a+5;

=> $a = 10 + 5 => $a= 15

**Bài tập 1**: Với $a = 20; $b = 5;

1. $a++ = 20. Nghĩa là $a = 20 xong mới tăng



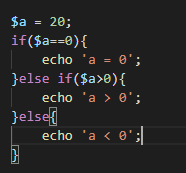


1. $a-- = 20
2. ++$a = 21
3. --$a = 19
4. $a++ + --$b - $b++ + ++$a + --$b = 20 + 4 – 4 + 22 + 4 = 46

# Câu điều kiện

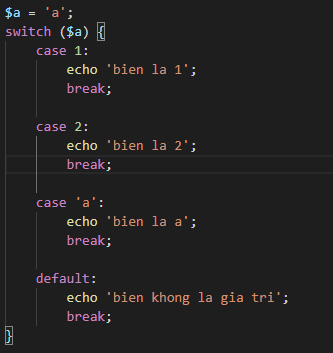
## If else

- Là câu lệnh rẽ nhánh điều kiện (nếu thì)



## Switch case

- Là câu lệnh rẽ nhánh có nhiều lựa chọn. Có thể thay thế if else tùy từng trường hợp.



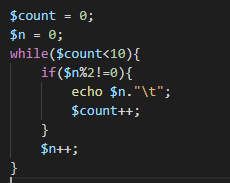
# Vòng lặp

## While

- Đây là vòng lặp

VD: In ra 10 số lẻ đầu tiên.

BG:



## Do While

- Là vòng lặp thực hiện hành động trước rồi sau đó mới kiểm tra điều kiện, sử dụng trong trường hợp muốn làm một menu

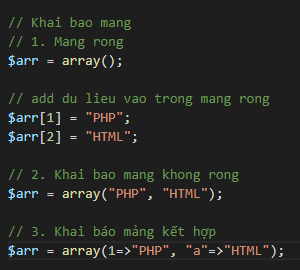
## For

- Là vòng lặp sẽ khai báo đầy đủ biến khởi tạo, điều kiện kết thúc, và tăng giảm biến trong cách viết câu lệnh.

# Mảng

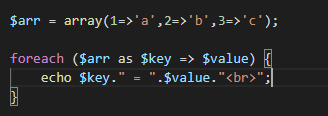
## Định nghĩa

- Là biến có nhiều phần tử lưu trữ dữ liệu trong đó



## Foreach

- [Vòng lặp foreach trong php](https://freetuts.net/vong-lap-foreach-trong-php-9.html) dùng để lặp các phần tử trong mảng, chính vì thế nó được sử dụng rất nhiều khi chúng ta làm dự án với PHP, hầu như ai cũng thích bởi vì sử dụng đơn giản



- Nó sẽ lấy toàn bộ các dữ liệu trong mảng

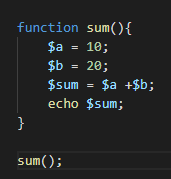
# Hàm

## Khái niệm

- Hàm là tổ hợp các đoạn mã nhằm để thực hiện một chức năng nào đó.

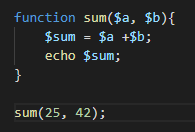
## Hàm tự định nghĩa không có tham số

VD:



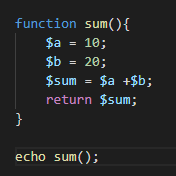
## Hàm tự định nghĩa có tham số

VD:



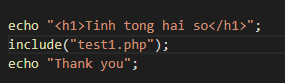
## Hàm có giá trị trả về

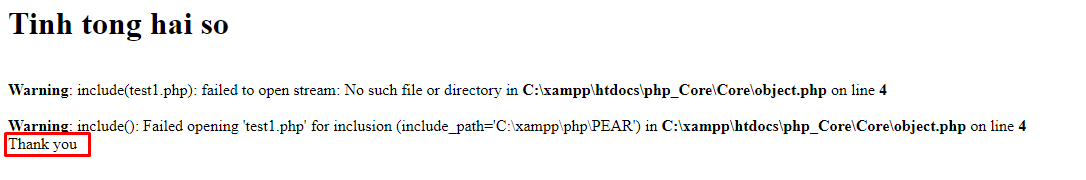
VD:



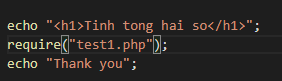
## Hàm triệu gọi file

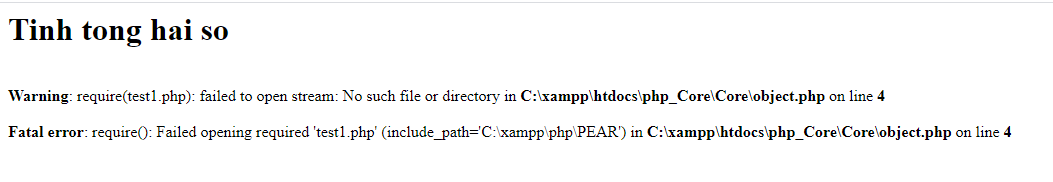
- Include(): Là hàm triệu gọi file vào trong chương trình code. Nếu gọi sai file hoặc file được gọi vào bị lỗi thì nó vẫn sẽ chạy tiếp những đoạn code phía sau và cảnh báo vấn đề gặp phải





- require(): Cũng là hàm triệu gọi file như include nhưng khi gọi sai file hoặc file được gọi vào bị lỗi thì nó sẽ dừng đoạn code lại ở chỗ bị lỗi. Và cảnh báo có lỗi



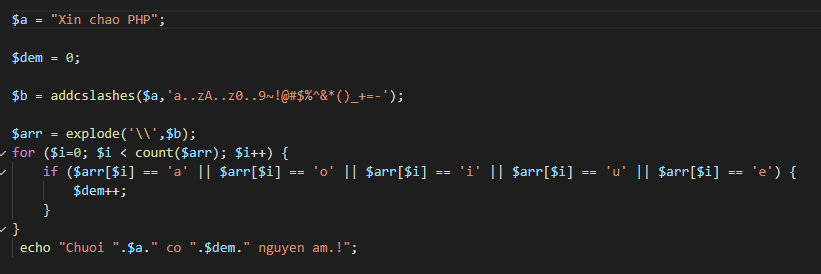


- require\_once() và Include\_once(): Cũng giống như require và include nhưng khi ta triệu gọi file vào 2 lần thì nó sẽ bỏ qua lần triệu gọi sau.

## Hàm có sẵn

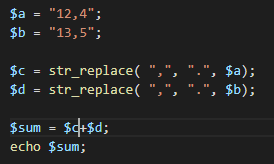
- Các hàm có sẵn hay dùng trong PHP là những hàm liên quan đến chuỗi và mảng

VD1: Kiểm tra một chuỗi xem trong đó có bao nhiêu nguyên âm.





VD2: Viết chương trình cho phép tính toán và in ra theo định dạng tiền tệ. Input1: 12,3 , Input2: 13,2 => Output: 25,5

# Kết nối Database

## Kết nối database với code php

$dbhost = "localhost";

$dbuser = "root";

$dbpass = "";

$dbname = "quanlysinhvien"; //Ten cua database

$conn = mysqli\_connect($dbhost, $dbuser, $dbpass, $dbname);

if ($conn) {

    mysqli\_query($conn, "'SET NAMES utf8'");

} else {

    echo "Ket noi that bai".mysqli\_connect\_error();

}

## Các câu lệnh truy vấn

// Bước 1: Triệu gọi file database

include\_once("test.php");

// Bước 2: Tiến hành truy vấn sql

$sql = " SELECT \* FROM `students` "; //Truy vấn dữ liệu

$query = mysqli\_query($conn, $sql); //Thực thi câu truy vấn

// Bước 3: Đếm số kết quả trả về

    // $total = mysqli\_num\_rows($query);

    // echo $total;

// Bước 4: Trả dữ liệu về dưới dạng mảng

    //$row = mysqli\_fetch\_array($query);

// Bước 5: In dữ liệu mảng ra màn hình

    while ($row = mysqli\_fetch\_array($query)) {

        echo $row["fullname"]."\t";

    }

# AJAX

- Để hiểu về AJAX ta cần có kiến thức cơ bản về HTML, Javascript, Xml,

- AJAX có thể trao đổi trực tiếp với máy chủ thông qua đối tượng XMLHttpRequest mà không cần tải lại trang.

## Làm việc với AJAX

Cú pháp:

<script>

    function ajaxFunction(){

    var xmlHttp;

    // Khoi tao doi tuong XMLHttpRequest

    if (window.XMLHttpRequest) {

        xmlHttp = new XMLHttpRequest();

    }else{

        xmlHttp = new ActiveXObject('Microsoft.XMLHTTP');

    }

    //Tiếp nhận dữ liệu

    xmlHttp.onreadystatechange = function(){

        if(xmlHttp.readyState == 4){

            document.getElementById('course').innerHTML=xmlHttp.responseText;

        }

    }

    //Gửi yêu cầu

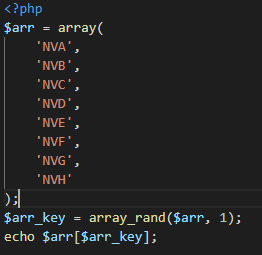
    xmlHttp.open('GET', 'ArrayAX.php', true);

    xmlHttp.send(null);

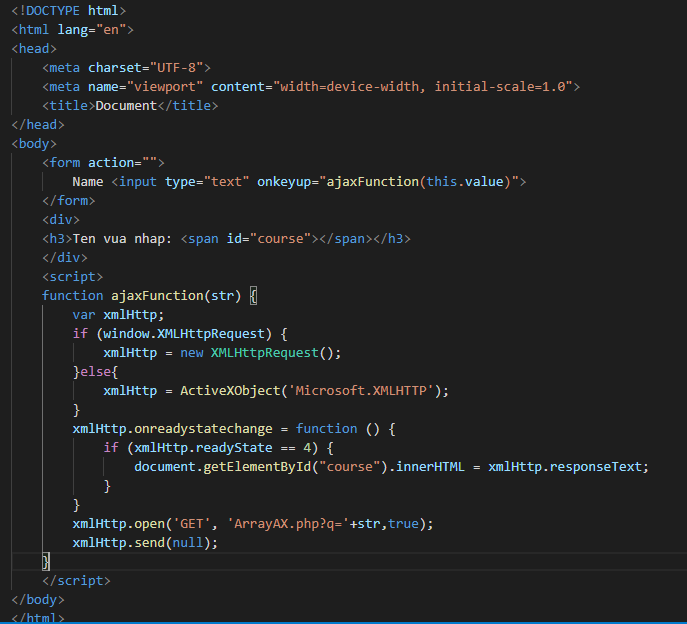
    }

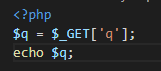
    </script>

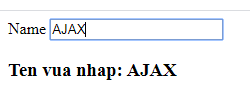
Với file ArrayAX.php như sau



Bài tập : Nhập vào ký tự trong ô input và nó sẽ tự động in ra chữ bên dưới mà không cần phải load lại trang bằng ajax







# JSON – XML

## JSON

- JSON là một cú pháp để lưu trữ và trao đổi dữ liệu dạng nhẹ

- JSON tương tự như XML nhưng cú pháp đơn giản hơn XML

$str = '[

{"class":"A", "name":"NVA"},

{"class":"B", "name":"NVB"},

{"class":"C", "name":"NVC"}

]';

//Cú pháp JSON đơn giản

$arr = json\_decode($str,true); //Đưa chuỗi json thành dạng mảng

echo $arr[0]['class']." ".$arr[0]['name']; //In mảng arr ra màn hình



## XML

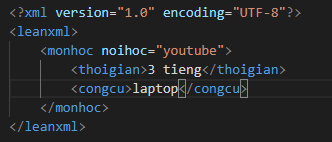
- XML là một ngôn ngữ đánh dấu.

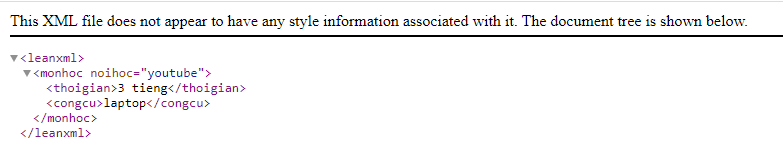
- XML có các cặp thẻ do người dùng tự định nghĩa

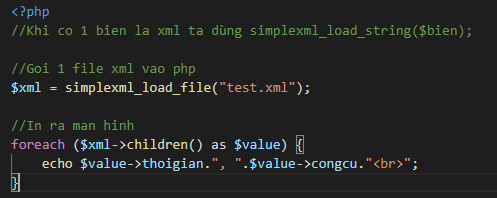
- XML là nơi để lưu trữ và vận chuyển dữ liệu

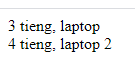
- XML là một cơ sở dữ liệu dạng nhỏ

VD: Tạo file test.xml





VD2: Truy xuất dữ liệu từ file test.xml bằng PHP  




VD3: Tạo một file test2.xml từ cơ sở dữ liệu đã có sẵn  
- Khi muốn cho người khác trích dẫn nguồn dữ liệu mà không ảnh hưởng đến dữ liệu trong Database thì ta sẽ tạo một file xml từ cơ sở dữ liệu.

// Ket noi toi co so du lieu

$dbhost = "localhost";

$dbuser = "root";

$dbpass = "";

$dbname = "quanlysinhvien";

$conn = mysqli\_connect($dbhost, $dbuser, $dbpass, $dbname);

if ($conn) {

    mysqli\_query($conn, "'SET NAMES utf8'");

} else {

    echo "Ket noi that bai".mysqli\_connect\_error();

}

$sql = "SELECT \*FROM students";

$query = mysqli\_query($conn,$sql);

//Luu co so du lieu vao chuoi

$text = '<?xml version="1.0" encoding="UTF-8"?>

<sinhvien>';

    while ($row = mysqli\_fetch\_array($query)) {

        $text .= '<thongtin>';

        $text .= '<fullname>'.$row['fullname'].'</fullname>';

        $text .= '<gender>'.$row['gender'].'</gender>';

        $text .= '<address>'.$row['address'].'</address>';

        $text .= '</thongtin>';

    }

$text .= '</sinhvien>';

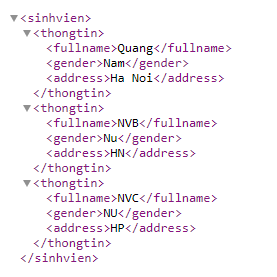
//Ghi ra file xml

$fp = 'test2.xml';

$fo = fopen($fp,"w+");

$fw = fwrite($fo,$text);

$fc = fclose($fo);

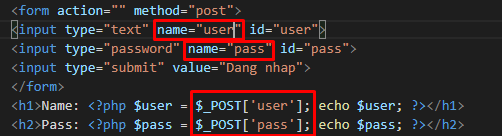


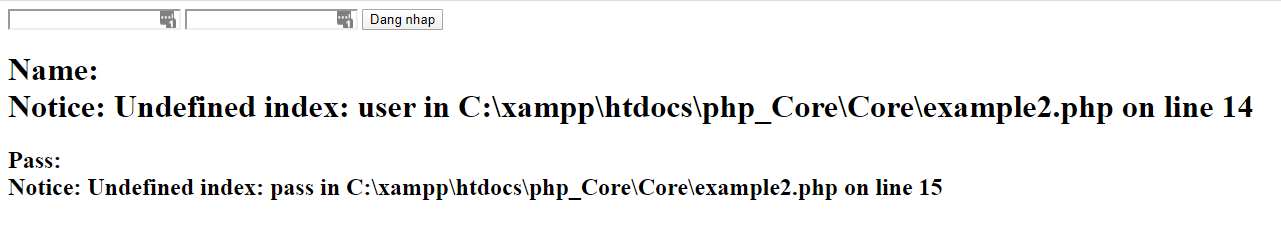
# Phương thức truyền dữ liệu POST – GET

## POST

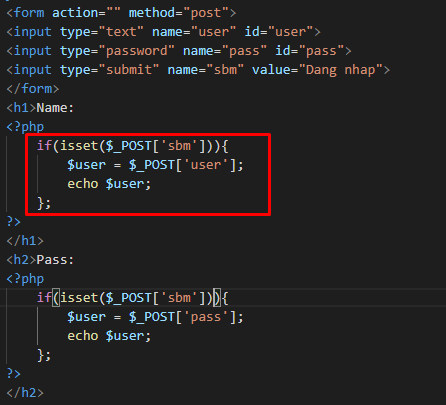
- Khi gửi dữ liệu từ form ra ngoài ta thường dùng các phương thức truyền dữ liệu là POST và GET. Trong đó phương thức POST là phương thức truyền dữ liệu bảo mật. Không làm lộ thông tin. Cho nên nó thường được dùng trong việc truyền nhận mật khẩu.

- Truyền biến bằng POST





- Khi chưa có dữ liệu truyền vào ta thấy browser sẽ cảnh báo lỗi như trên. Để khắc phục lỗi này ta làm như sau





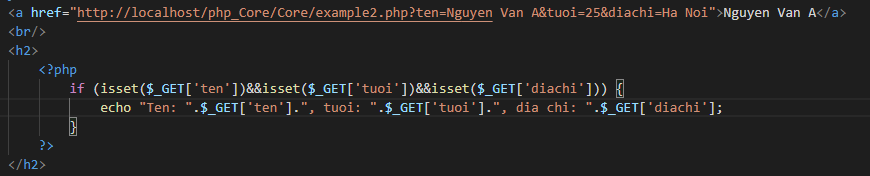
- Với đoạn code trên ta phải kiểm tra sự tồn tại của nút submit. Nhưng câu điều kiện if else chỉ nhận giá trị là true hoặc false. Và để giải quyết vấn đề này ta có hàm **isset()**. Nếu biểu thức trong hàm isset() tồn tại nó sẽ trả về là true, ngược lại nó sẽ trả về là false.

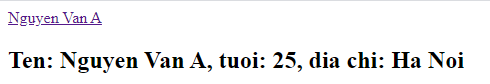
## GET

- Cách khai báo và ý nghĩa tương tự như phương thức POST nhưng giá trị và biến được thể hiện trên thanh url dưới cú pháp: filename.php?var1=value1&var2=value2&var3=value3

- Với phương thức này nó không cần phải khai báo form mà truyền trực tiếp giá trị cho biến cũng được.

- Khi ta đã sử dụng form thì method của form là get thì ta phải sử dụng phương thức get để lấy giá trị và ngược lại với post.





# SESSION – COOKIE

## SESSION

- Là một phiên làm việc của người dùng trên website

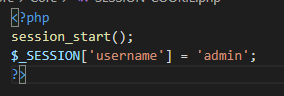
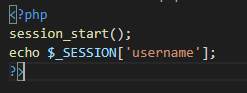
- SESSION bắt đầu khi người dùng làm việc với website. Kết thúc khi tắt trình duyệt hoặc hủy bằng câu lệnh.

Cú pháp:

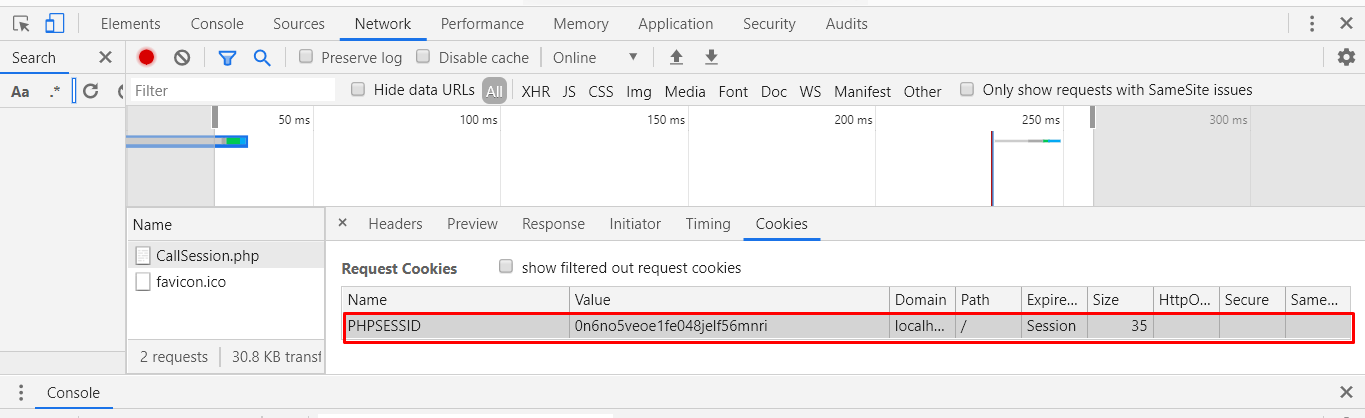
+ Khai báo phiên làm việc: session\_start();. Nó luôn luôn nằm trên đầu của các trang website cần có session

+ Khởi tạo giá trị cho sesion: $\_SESSION['username'] = 'admin';

VD: Khởi tạo một session





+ Hủy session bằng câu lệnh: session\_unset('username'); hoặc session\_destroy();

## COOKIE

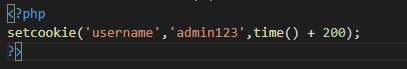
- Giống khái niệm session là gì, khái niệm **cookie** cũng được dùng để lưu những thông tin tạm thời. Nhưng tập tin cookie sẽ được truyền từ server tới browser và được lưu trữ trên máy tính của bạn khi bạn truy cập vào ứng dụng.

- Cú pháp khai báo: setcookie('name','value','time');

- Cú pháp sử dụng: $\_COOKIE['name'];

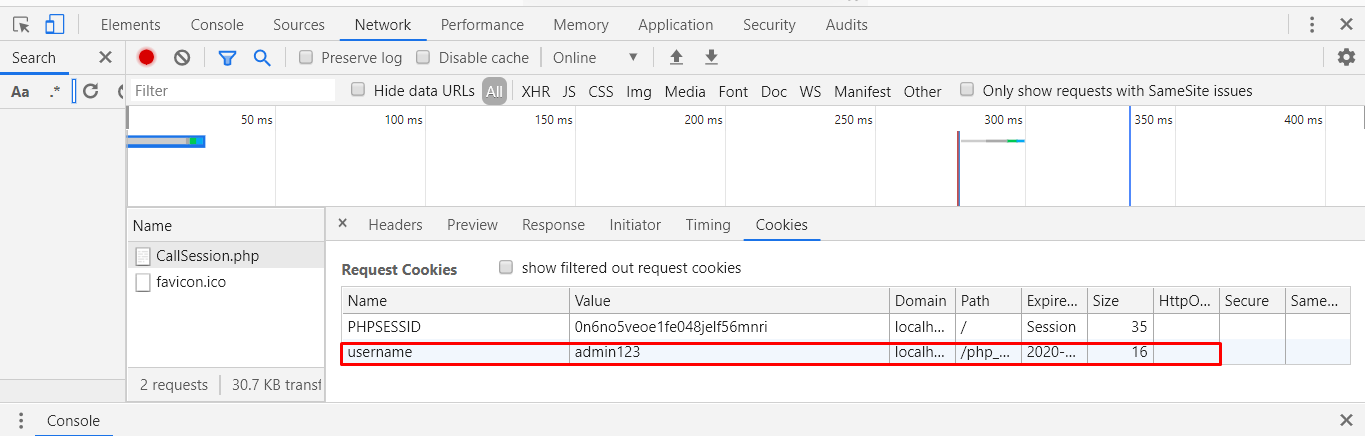
- Cú pháp hủy: setcookie('username'); hoặc setcookie('username','admin123',time() - 200);

VD: Tạo một cookie đơn giản









| **Cookie** | **Session session là gì session là gì** |
| --- | --- |
| **Cookie** được lưu trữ trên trình duyệt của người dùng. | **Session** không được lưu trữ trên trình duyệt. |
| Dữ liệu **cookie** được lưu trữ ở phía client. | Dữ liệu **session** được lưu trữ ở phía server. |
| Dữ liệu **cookie** dễ dàng sửa đổi hoặc đánh cắp khi chúng được lưu trữ ở phía client. | Dữ liệu **session** không dễ dàng sửa đổi vì chúng được lưu trữ ở phía máy chủ. |
| Dữ liệu **cookie** có sẵn trong trình duyệt đến khi expired. | Sau khi đóng trình duyệt sẽ hết phiên làm việc (session) |