CSDL & Web

PHP

Giáo viên: Lương Hán Cơ
Nhóm thực hiện
0512275 – Trần Duy Quang
0512375 – Mai Anh Tuấn
0512387 – Lê Anh Tài
0512396 – Hoàng Anh Tú

NỘI DUNG CHÍNH

- GIỚI THIỆU PHP
- HƯỚNG ĐỐI TƯỢNG TRONG PHP
- REGULAR EXPRESSION

GIỚI THIỆU PHP

NỘI DUNG

- Giới thiệu PHP
- Cú pháp
- Quy tắc chung
- Biến & Hàm

GIỚI THIỆU PHP

PHP **H**ypertext **P**reprocessor

- Ngôn ngữ kịch bản phía **server**
- Tập tin có phần mở rộng **.php**
- PHP có cú pháp ngôn ngữ giống **C** & **Perl**
- Phiên bản mới nhất là **5.3** alpha 1 (T8/2008)
  www.php.net

ĐẶT MÃ LỆNH

- Cú pháp chèn mã lệnh php

Mở	Đóng
<?php	?>
<script language="php">	<script>
<?	?>
<%	%>

NGẮT LỆNH

```
<?php
    $i = 0;
    while ($i < 10) {
?>
            Current i is:
<?php
            echo $i;
    }
?>
```

QUY TẮC CHUNG

- Khối lệnh được bao trong dấu {}
- Mỗi lệnh kết thúc bằng dấu ;
- Các lệnh là một trong các lệnh sau:
  - Điều kiện: if… else…, switch
  - Lặp : while, do while, for, foreach, with
  - Thao tác đối tượng
- Cách ghi chú thích:

```
// Chú thích 1 dòng
# Chú thích 1 dòng
/* Chú thích
   nhiều dòng */
```

KIỂU DỮ LIỆU

- boolean (bool)
- **integer** (int)
- **double** (float, real)
- string
- **array**
- **object**
- resource: tham chiếu đến tài nguyên

Biến trong PHP có thể lưu **mọi** kiểu dữ liệu

PSEUDO TYPE

- mixed: tham số có thể chấp nhận nhiều kiểu dữ liệu
- number: có thể là integer hoặc float
- callback

BIẾN

- Bắt đầu bằng **$**
- **Không** khai báo kiểu
- Tự động khởi tạo ở lần gán giá trị đầu tiên

  $tênBiến = giáTrị;
- [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*
- **Phân biệt** hoa thường

BIẾN CỦA BIẾN

```
$a = 'hello';
$$a = 'world';
```

Có **2** biến

Một biến $a giá trị 'hello'

Một biến $hello giá trị 'world'

PHẠM VI CỦA BIẾN

Biến toàn cục
Biến toàn cục
Biến tĩnh
Biến cục bộ

```
<?php
    $a = 0;
    function Test(){
        global $a;
        static $c = 0;
        $b = $a +1;
    }
?>
```

BIẾN TOÀN CỤC ĐỊNH NGHĨA SẴN

- $_SERVER
- $_REQUEST
- $_GET
- $_POST
- $_SESSION

CHUYỂN KIỂU

- Tự động

```
$x = 0;
$x = 100 + "nam";
```

- Ép kiểu

```
$x = (int) 32.5;
```

- Dùng hàm

```
settype($x, "int");
```

HẰNG SỐ

- Định nghĩa:

```
define("MAX", 100);
```

- Chỉ có thể có kiểu boolean, integer, float, string
- Nên tránh đặt **magic constant** (__MAX__)
- Gọi hàm bằng hằng

```
define("HamTest", "Test");
call_user_func(HamTest);
```

CẤU TRÚC ĐIỀU KHIỂN - Rẽ nhánh

- if, if..else
- if (i == 0):
  i++;
  elseif (i == 1):
  i += 2;
  else:
  i += 3;
  endif;

CẤU TRÚC ĐIỀU KHIỂN – Lặp

- while (i > 0)
  i++;
- while (i>0):
  i++;
  endwhile;
- do{
  i--;
  } while (i > 0);

CẤU TRÚC ĐIỀU KHIỂN – Lặp

```
$a = array(1, 2, 3);
foreach ($a as $i)
   $i *= 2;

switch ($i){
   case 0:
   break;
}
```

HÀM

```
function TenHam(ThamSo1, thamSo2,…)
{
}

function TenHam(ThamSo1, ThamSo2, …)
{
   return 0;
}
```

HÀM ĐIỀU KIỆN

```
$flag = TRUE;
if($flag){
   function Test(){
        echo "Hello";
   }
}
if ($flag) Test();
```

HÀM TRONG HÀM

```
function A(){
   function B(){
   }
}

A();
B();   //Có thể gọi B do
       //gọi A làm B tồn tại
```

ĐỐI SỐ CỦA HÀM

```
• function Test( $a,
                 &$b,
                 $c = 0)
{
}
```

Tham trị

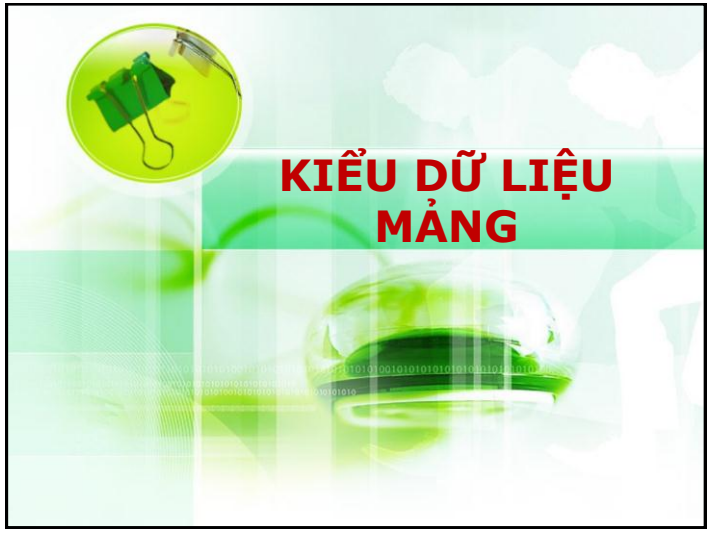
Tham chiếu

Đối số mặc định

HÀM BIẾN

```
function Test(){
}

$func = 'Test';
$func(); //Gọi hàm Test
```



NỘI DUNG

- Mảng một chiều
- Mảng nhiều chiều

KHAI BÁO MẢNG

- Dùng dấu []

```
$a[] = ‘sv1’;
```

- Dùng dấu [] với thuộc tính khóa

```
$a[‘second’] = 2;
```

- Với từ khóa array

```
$a = array(‘sv1’, 15, 3.2);
```

MẢNG CÓ GIÁ TRỊ KẾT BUỘC

```
$a = array( "first"=>1,
            "second”, // key = 0
          "third"=>3.0);
echo $a[0];// “second”
```

Kết buộc từng Key với Value

Chỉ định chỉ số bắt đầu của mảng

```
$a = array(10 => “ten”,
           “eleven”,
           “twelve”);
echo $a[11]; //“eleven”
```

TRUY XUẤT MẢNG CÓ GIÁ TRỊ KẾT BUỘC

```
$a = array(     10,
                5   => “five”,
                3   => “three”,
                3.2,
                ‘8’  => 5,
                ‘02’ => 7.7,
                0    => 12)
echo $a[0];
```

Key = 0

Key = 6

12

THAO TÁC VỚI MẢNG

- Thêm phần tử động

```
$a[] = ‘sv1’;
$a[] = ‘sv2’;
```

- Loại phần tử khỏi mảng

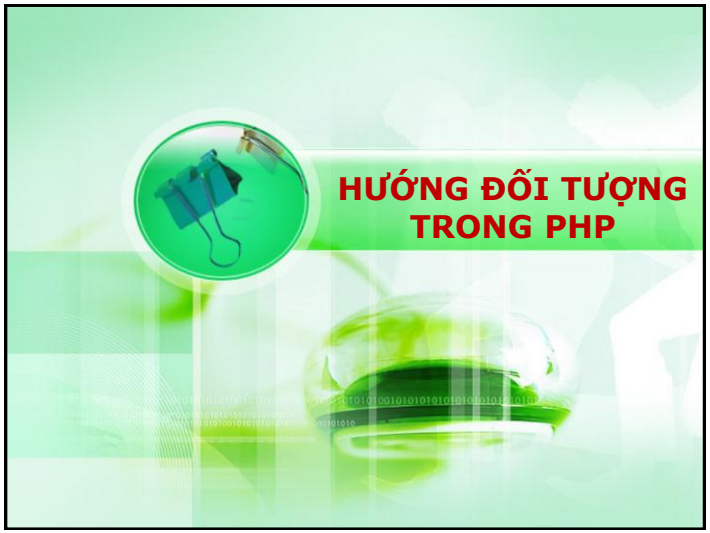
```
unset($a[1]);
```

- Xóa toàn bộ mảng

```
unset($a);
```

MẢNG NHIỀU CHIỀU

```
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                       "b" => "banana",
                                       "c" => "apple"
                                     ),
                  "numbers" => array ( 1, 2, 3, 4, 5,6),
                  "holes"   => array (  "first",
                                        5 => "second",
                                        "third"
                                     )
                );
```



NỘI DUNG

- Lớp trong PHP
- Kế thừa

KHAI BÁO LỚP

Hằng số của lớp
Biến tĩnh
Hàm tĩnh
Hàm Tạo và Hủy
Biến thành phần
Hàm thành phần

```
class DaGiac{
    const DATA = "abc";
    public static $Count;
    public static function KiemTra(){}
    public function __construct(){}
    public function __destruct(){}
    private $ten;
    public function getTen(){
        return $this->ten;
    }
}
```

SỬ DỤNG LỚP

```
$dg = new DaGiac;
echo DaGiac::DATA;
echo DaGiac::Count;
echo $dg->getTen();
```

LỚP TRỪU TƯỢNG

```
abstract class AbstractClass
{
    //Lớp con phải cài lại 2 hàm này
    abstract protected function getValue();
    abstract protected function setValue($prefix);

    // Phương thức bình thường
    public function printOut() {
        print $this->getValue() . "\n";
    }
}
```

GIAO DIỆN

```
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}
```

KẾ THỪA

- Dùng từ khóa extends
- Không đa kế thừa lớp
- Đa kế thừa giao diện

VÍ DỤ

```
class A extends B implements C, D, E{
}
```

THÊM ĐỘNG THUỘC TÍNH CHO LỚP

Thêm thuộc tính

Lấy giá trị

```
Class DynamicAttrib{
        private $data = array();

        public function __set($name, $value) {
            $this->data[$name] = $value;
        }

        public function __get($name) {
        if (array_key_exists($name, $this->data))
                return $this->data[$name];
        }
}

$dyn = new DynamicAtrib;
$dyn->a = 1;
echo $dyn->a;
```

GIỚI THIỆU

- Là 1 tập các ký hiệu dùng để miêu tả 1 tập các chuỗi
- Có thể ứng dụng để phân tích cú pháp chuỗi, tìm kiếm và kiểm tra chuỗi
- Xử lý chậm hơn các hàm xử lý chuỗi đơn giản
- PHP hỗ trợ 2 loại regular expression: POSIX-extended và Perl-Compatible Regular Expressions (PCRE)

QUI TẮC CHUNG

- Hầu hết mọi ký tự đều dùng để biểu diễn chính nó
- Ví dụ:
  - Chuỗi "foo" sẽ tìm thấy trong câu "John plays football"
- Ngoài ra cũng có 1 số ký đặc biệt tự mang ý nghĩa riêng

CÁC KÝ TỰ ĐẶC BIỆT

- Các ký tự đặc biệt bao gồm: ([{\^-$|]})?*+.
- Ký tự ^:
  - Đứng đầu 1 regular expression
  - Chuỗi tương ứng phải nằm ở phần đầu trong chuỗi cần tìm
  - Ví dụ:
    - "^foo " sẽ được tìm thấy trong "food"
    - Sẽ không tìm thấy trong "dfoo"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự $:
  - Đứng cuối 1 regular expression
  - Chuỗi tương ứng phải nằm cuối trong chuỗi cần tìm
  - Ví dụ:
    - "foo$ " sẽ được tìm thấy trong chuỗi "dfoo"
    - Không tìm thấy trong chuỗi "food"
    - "^foo$" sẽ tìm thấy trong chuỗi "foo" nhưng không tìm thấy trong "food" hay "dfoo"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự .:
  - Tương ứng với 1 ký tự bất kỳ trừ ký tự xuống dòng \
  - Ví dụ:
    - "h.t" sẽ được tìm thấy trong chuỗi "hat", "hot"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự |:
  - Tương tự như toán tử OR
  - Mang ý nghĩa chuỗi cần tìm có thể ứng với 1 trong các biểu thức được nối bằng dấu |
  - Ví dụ:
    - "aaa|bbb" có thể tương ứng với các chuỗi như "aaa", "bbb" hay "aaabbb"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự ():
  - Dùng để gom nhóm
  - Ví dụ:
    - "(a|b)bb" có thể tương ứng với các chuỗi như "abb", "bbb"
    - "(gif|jpg) " có thể tương ứng với các chuỗi như "gif" hoặc "jpg"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự \:
  - Dùng để vô hiệu ý nghĩa đặc biệt của các ký tự đặc biệt
  - Ví dụ:
    - "\^a" có thể tương ứng với chuỗi "^a"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự +:
  - Xác định lặp với số lượng ít nhất là 1
  - Ví dụ:
    - "a+" tương ứng với chuỗi "a", "aa", "aaa",…
- Ký tự *:
  - Xác định lặp với số lần ít nhất là 0
  - Ví dụ:
    - "ab*" tương ứng với chuỗi "a", "ab", "abb",…

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự ?:
  - Có tồn tại hay không tồn tại
  - Ví dụ:
    - "ab?" tương ứng với chuỗi "a", "ab"

CÁC KÝ TỰ ĐẶC BIỆT

- * cũng có thể được dùng để kiểm tra tồn tại hay không tồn tại như ? nhưng giá trị chuỗi tìm thấy sẽ khác nhau
  - Ví dụ:
    - Chuỗi "abb"
    - Biểu thức "ab*" tìm thấy giá trị là "abb"
    - Biểu thức "ab?" tìm thấy giá trị là "ab"

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự {}:
  - Có thể có nhiều cách sử dụng
  - {n}: lặp n lần
  - Ví dụ:
    - "a{3}" tương ứng với "aaa"

CÁC KÝ TỰ ĐẶC BIỆT

- {m,}: lặp ít nhất m lần
- Ví dụ:
  - "a{2,}" tương ứng với "aa", "aaa",…
- {m,n}: lặp từ m tới n lần
- Ví dụ:
  - a{2,3} tương ứng với aa, aaa

CÁC KÝ TỰ ĐẶC BIỆT

- Ký tự []:
  - Dùng tạo ra các lớp trong 1 biểu thức
  - 1 lớp có thể biểu diễn 1 dãy các ký tự như là 1 ký tự trong biểu thức
  - Ví dụ:
    - "[abc]" tương ứng với "a", "b" hoặc "c"
    - "[a-z] " tương ứng với tất cả các ký tự chữ cái thường
    - "[^A-Za-z0-9] " tương ứng với tất cả các ký tự khác chữ cái hoặc số
    - "([wx])([yz]) " tương ứng với "wy", "wz", "xy", "xz"

CÁC KÝ TỰ ĐẶC BIỆT

- Ngoài ra còn 1 số cách biểu diễn khác
  - \d tương đương với [0-9]
  - \D tương đương với [^0-9]
  - \w tương đương với [a-zA-Z0-9_]
  - \W tương đương với [^a-zA-Z0-9_]
  - \s tương đương với [\r\t\n\f]
  - \S tương đương với [^ \r\t\n\f]

CÁC HÀM LIÊN QUAN TRONG PHP

- Bao gồm 2 loại: POSIX và PCRE
- Các hàm của PCRE mạnh và nhanh hơn của POSIX
- PCRE là các hàm sử dụng tương thích với Pearl nên 1 biểu thức phải đặt bắt đầu và kết thúc bằng dấu /

CÁC HÀM CỦA POSIX

- Hàm ereg($re, $text, $array)
  - Dùng để so khớp 1 biểu thức trong 1 chuỗi
  - Tham số:
    - $re: là biểu thức regular expression
    - $text: là chuỗi cần tìm kiếm
    - $array: là mảng chuỗi đầu tiên được so khớp với $re trong $text
  - Giá trị trả về:
    - Số nguyên: số chữ cái khớp $re trong $text
    - false: nếu không tìm thấy chuỗi con nào khớp với $re

CÁC HÀM CỦA POSIX

- Ví dụ:

$a= ereg("aa","aabbb",$array);

  - $a = 2;
  - $array gồm 1 phần tử với:
    - $array[“0”] = aa;

CÁC HÀM CỦA POSIX

- Hàm egrep_replace($re, $sub, $text)
  - Dùng để thay thế tất cả các chuỗi con trong $text khớp với biểu thức $re bằng chuỗi $sub
  - Tham số:
    - $re: biểu thức regular expression
    - $sub: chuỗi con được thay thế
    - $text: chuỗi gốc
  - Giá trị trả về: chuỗi $text sau khi đã được thay thế

CÁC HÀM CỦA POSIX

- Ví dụ:

  $a = ereg_replace("a", "b", "abababab");

  – $a = “bbbbbb”
- Hàm ereg_replace phân biệt hoa thường.
- Sử dụng hàm eregi_replace để không phân biệt hoa thường

CÁC HÀM CỦA POSIX

- Có thể thay thế thứ tự các nhóm trong 1 biểu thức ở chuỗi kết quả khi thay thế
- Gom nhóm bằng ký tự ()
- Ví dụ:

$test = "25/12/2000";

$a=ereg_replace("([0– 9]+)/([0– 9]+)/([0– 9]+)", "\\2/\\1/\\3", $test);

– $a = “12/25/2000”

CÁC HÀM CỦA POSIX

- Hàm split($re, $text)
  – Dùng để cắt $text thành các chuỗi con phân tách với nhau bởi các chuỗi khớp với biểu thức $re
  – Tham số:
    - $re: biểu thức regular expression
    - $text: chuỗi gốc
  – Giá trị trả về: mảng các chuỗi con sau khi đã cắt

CÁC HÀM CỦA POSIX

- Ví dụ:

  $text = "apples, oranges, peaches and grapefruit";

  $fruitarray = split(", | and ", $text);

  – fruitarray[“0”] = “apples”

  – fruitarray[“1”] = “oranges”

  – fruitarray[“2”] = “peaches”

  – fruitarray[“3”] = “grapefruit”

CÁC HÀM CỦA PCRE

- Sử dụng cú pháp giống như Pearl
- Phải phân tách 1 biểu thức bằng ký tự phân cách, thông thường là /
- Các hàm sử dụng tốt hơn và nhanh hơn so với các hàm POSIX

CÁC HÀM CỦA PCRE

- Hàm preg_match($re, $text, $array)
  - Dùng để so khớp biểu thức trong 1 chuỗi
  - Tham số:
    - $re: là biểu thức regular expression
    - $text: là chuỗi cần tìm kiếm
    - $array: là mảng chuỗi đầu tiên được so khớp với $re trong $text
  - Giá trị trả về:
    - 1: nếu có chữ cái so khớp trong $text
    - 0: nếu không tìm thấy chuỗi con nào khớp với $re

CÁC HÀM CỦA PCRE

- Ví dụ:

  $a = preg_match("/h.t/", "hat", $array);
  - $a = 1
  - $array[“0”] = “hat”
- Nếu dùng hàm ereg thì $a = 3

CÁC HÀM CỦA PCRE

- Thông thường, preg_match luôn so khớp số phần tử nhiều nhất có thể nhưng ta có thể lấy chuỗi khớp ít nhất
- Ví dụ:

$text = "pot post pat patent";

$a = preg_match(“/h.*?t/ ", “hat t”, $array);

- $a = 3
- $array[“0”] = “hat”

CÁC HÀM CỦA PCRE

- Hàm preg_match_all($re, $text, $array)
  - Dùng để so khớp và lấy tất cả các chuỗi con khớp với $re trong $text
  - Tham số:
    - $re: là biểu thức regular expression
    - $text: là chuỗi cần tìm kiếm
    - $array: là mảng các chuỗi được so khớp với $re trong $text
  - Giá trị trả về:
    - Số nguyên: số lượng chuỗi so khớp
    - 0: nếu không tìm thấy chuỗi con nào khớp với $re

CÁC HÀM CỦA PCRE

- $array là 1 mảng 2 chiều chứa các chuỗi đã khớp với biểu thức:
  - Phân tử đầu tiên chứa các chuỗi so khớp hoàn toàn với biểu thức
  - Các phân tử còn lại chứa các chuỗi so khớp với từng nhóm của biểu thức

CÁC HÀM CỦA PCRE

- Ví dụ:

$text = "01-05-99, 02-10-99";

$a = preg_match_all("/(\d+)-(\d+)/", $text, $array);

  - $a = 2
  - $array[“0”][“0”] = “01-05”
  - $array[“0”][“1”] = “02-10”
  - $array[“1”][“0”] = “01”
  - $array[“1”][“1”] = “02”
  - $array[“2”][“0”] = “05”
  - $array[“2”][“1”] = “10”

CÁC HÀM CỦA PCRE

- Hàm preg_replace($re, $sub, $text)
  - Thay thế các chuỗi con khớp với $re trong $text bằng chuỗi $sub
  - Tham số:
    - $re: biểu thức regular expression
    - $sub: chuỗi con dùng để thay thế
    - $text: chuỗi gốc
  - Giá trị trả về: chuỗi sau khi thay thế các phần tử khớp

CÁC HÀM CỦA PCRE

- Hàm preg_replace($re, $sub, $text)
  - Thay thế các chuỗi con khớp với $re trong $text bằng chuỗi $sub
  - Tham số:
    - $re: biểu thức regular expression
    - $sub: chuỗi con dùng để thay thế
    - $text: chuỗi gốc
  - Giá trị trả về: chuỗi sau khi thay thế các phần tử khớp

CÁC HÀM CỦA PCRE

- Ví dụ:

```
$t = "25/12/99";
$a= preg_replace( "|(\d+)/(\d+)/(\d+)|", "\\2/\\1/\\3", $t );
```

  - $a = "12/25/99 "