Lập trình PHP

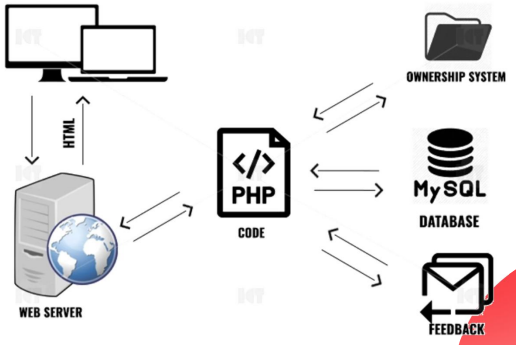
Su Kim Anh

PHP

1. Giới thiệu ngôn ngữ lập trình PHP
2. Một số khái niệm
3. Khai báo và gán giá trị cho biến
4. Phạm vi hoạt động của biến
5. Xuất dữ liệu ra trình duyệt
6. Kiểu dữ liệu trong PHP
7. Cấu trúc điều khiển trong PhP
8. Hàm
9. Cơ chế truyền nhận dữ liệu

Giới thiệu ngôn ngữ lập trình PHP

- Ngôn ngữ lập trình PHP (Hypertext Preprocessor)
  - Là ngôn ngữ lập trình phía server-side được thiết kế để xây dựng và ứng dụng web



Giới thiệu ngôn ngữ lập trình PHP

- Thiết lập trang PHP
  - Là một trang động (Dynamic Page)
  - Có charset = UTF-8 (để trang hiển thị tiếng Việt)
  - Nhúng code PHP vào trang bằng thẻ php
  - Hiển thị nội dung trên trang bằng lệnh *echo*

Giới thiệu ngôn ngữ lập trình PHP

- Quy ước
  - Code PHP được đặt trong các thẻ sau:

Thẻ mở	Thẻ đóng
<?php	?>
<?	?>
<script language="php">	</script>

  - Ví dụ:

```
<?php echo "<b>Chào các bạn.</b>" ?>
<? echo "<b>Chào các bạn.</b>" ?>
<script language="php">echo "<b>Chào các bạn.</b>"</script>
```

Giới thiệu ngôn ngữ lập trình PHP

- Quy ước
  - Các lệnh kết thúc bằng dấu ;
  - Ghi chú trong PHP:

Các kiểu dữ liệu cơ sở

- Đặc điểm
  - Kiểu Boolean: chỉ có một trong hai giá trị là TRUE và FALSE
  - Kiểu Integer (số nguyên): Giá trị có thể là số trong hệ thập phân, thập lục phân và bát phân.
    - Ví dụ:
      - 1234 // hệ thập phân
      - -123 // số âm hệ thập phân
      - 0123 // hệ bát phân (bắt đầu bằng 0)
      - 0x1A // hệ thập lục phân (bắt đầu bằng 0x)

Các kiểu dữ liệu cơ sở

- Đặc điểm
  - Kiểu Float/ Double (số thực)
    - Ví dụ:
      - 1.234
      - 1.2e3 → 1.2 * 1000 = 1200
  - Kiểu String (chuỗi, ký tự)
    - Mỗi ký tự chiếm 1 byte
    - Mỗi chuỗi có thể chứa một hay nhiều ký tự thuộc 256 ký tự khác nhau
    - Chuỗi không có giới hạn về kích thước

Các kiểu dữ liệu cơ sở

- Đặc điểm
  - Kiểu Array (mảng các phần tử)
    - Ví dụ:
      - array(1, 2, 3, 4, 5)
  - Kiểu Object (đối tượng)
    - Ví dụ
      - `$xe_hoi = new Xe(); // đối tượng xe hơi`

Biến

- Khai báo
  - Cú pháp: $ten_bien
- Quy tắc đặt tên biến
  - Bắt đầu bằng ký tự $, theo sau là một ký tự hoặc dấu _, tiếp đó là ký tự, ký số hoặc dấu _
  - Nên khởi tạo giá trị ban đầu cho biến
  - Không trùng với tên hàm
  - Không nên bắt đầu bằng ký số

*Lưu ý: Tên biến có phân biệt chữ HOA – chữ thường

Biến

- Gán giá trị cho biến
  - Trong PHP, khi khai báo Biến không cần ghi kiểu mà nó tự xác định kiểu dữ liệu cho biến khi biến được gán giá trị.

Khai báo chuỗi Ho_ten với giá trị Lê hùng	$Ho_ten = "Lê Hùng"; // String
Khai báo số nguyên m với giá trị 4	$m = 4; // Integer
Khai báo số thực y với giá trị là 3,4	$y = 3.4 ; // Double
Khai báo biến logic Gioi_tinh với giá trị Nam	$Gioi_tinh = true;

Biến

❑ Kết xuất

Vấn đề:
Cần kết xuất thông tin của biến x
Ví dụ:
Kết xuất biến chuỗi Ho_ten
Kết xuất biến số thực y
Kết xuất biến logic Gioi_tinh
Giải quyết:
Khai báo chuỗi $Chuoi với giá trị ban đầu tương ứng ghi chú (hằng chuỗi) về nội dung của biến cần xuất
Bổ sung vào $Chuoi giá trị của biến cần xuất
Xuất $Chuoi

```
// Kết xuất Ho_ten
$Chuoi = "Ho ten:";
$Chuoi = $Chuoi . $Ho_ten;
// Kết xuất y
$Chuoi = "y = ";
$Chuoi = $Chuoi . $y;
// Kết xuất Gioi_tinh
$Chuoi = "Gioi tinh: ";
$Chuoi = $Chuoi + $Gioi_tinh;
```

Lệnh kết xuất chuỗi
Ý nghĩa : Cho phép xuất thông tin của biến hay hằng chuỗi
Cú pháp
echo Hằng;
Hay
echo Biến;

Phép toán bổ sung chuỗi
Ý nghĩa : Cho phép bổ sung thông tin của biến hay hằng chuỗi vào biến chuỗi $S
Cú pháp
$S = $S . Hằng chuỗi
Hay
$S = $S + Biến chuỗi

Biến

- Nhận giá trị từ điều khiển

Vấn đề:
Cần nhận giá trị từ điều khiển và gán cho biến x
Ví dụ:
Chuỗi Ho_ten nhận giá trị từ điều khiển textfield txtHo_ten
Biến y nhận giá trị từ điều khiển textfield txtY
Biến logic Gioi_tinh nhận giá trị từ điều khiển radio button rdoGioi_tinh
Giải quyết:
x = $_POST["Tên điều khiển tương ứng"

```
// Nhận Ho_ten
$Ho_ten = $_POST["txtHo_ten"];
// Nhận y
$y = $_POST["y"];
// Nhận Gioi_tinh
$Gioi_tinh =
$_POST["rdoGioi_tinh"]
```

Biến

- Các phương thức kiểm tra giá trị của biến

Vấn đề:
Cần kiểm tra xem biến x có giá trị hay không, kiểu dữ liệu của biến x...
Ví dụ:
- Kiểm tra sự tồn tại của biến Ho_ten
- Kiểm tra biến y xem có rỗng hay không
- Kiểm tra biến m có phải là kiểu số hay không
- Kiểm tra biến y có phải là kiểu double hay không
- Xác định kiểu của biến Ho_ten

Giải quyết:
Sử dụng các phương thức kiểm tra giá trị của biến

```
// Kiểm tra tồn tại của biến Ho_ten
$kq = isset($Ho_ten); // true hoặc false
// Kiểm tra biến y có rỗng hay không
$Is_empty = empty($y); // true hoặc false
// Kiểm tra biến m có phải là kiểu số hay không
$Is_number = is_numeric($m); // true hoặc false
// Kiểm tra biến y có phải là kiểu double hay không
$Is_y_double = is_double($y); // true hoặc false
// Xác định kiểu của biến Ho_ten
$Type_Ho_ten = gettype($Ho_ten); // string
```

Hằng

- Hằng là một giá trị không thể chỉnh sửa trong quá trình thực hiện chương trình
  - Ví dụ: Pi, tỷ giá..
- Hằng là một giá trị không thể chỉnh sửa trong quá trình thực hiện chương trình
  - Quy tắc đặt tên hằng cũng giống như quy tắc đặt tên biến
  - Tin hằng thường IN HOA

Hằng

- Khai báo
  - Dùng hàm define() để định nghĩa

```
define("TÊN_HẰNG", giá_trị);
```

  - Ví dụ: Khai báo hằng số PI có giá trị là 3.14

```
define("PI", 3.14);
```

  - Ghi chú
    - Một khi hằng được định nghĩa, nó không bị thay đổi.
    - Chỉ có các kiểu dữ liệu boolean, integer, float, string mới có thể chứa các hằng.

HÀM

- Phân loại
  - *Hàm do PHP cung cấp*. Có trên 700 hàm chia thành nhiều nhóm: Chuỗi, toán, thời gian, lịch, mảng, tập tin, mail, xử lý CSDL,... chỉ cần gọi hàm khi sử dụng và truyền vào các giá trị phù hợ dựa trên các tham số
  - *Hàm do người dùng định nghĩa*
    - Ví dụ:

```
//ax + b = 0
$Nghiem =  Giai_Phuong_Trinh_Bac_I(2,4);
```

HÀM

Phương thức xây dựng hàm

❑Khai báo

```
function
Tên_hàm(Danh_sách_các_tham_số)
{
   khối lệnh bên trong hàm
   return giá_trị;
}
```

- Trong đó:
  - Tên hàm: được sử dụng khi gọi hàm, tên hàm nên có ý nghĩa gợi nhớ
  - Danh sách các tham số: dùng để truyền dữ liệu bên ngoài vào, hàm có thể có hoặc không có tham số
  - Giá trị: là kết quả trả về của hàm. Hàm có thể có hoặc không có giá trị trả về

Sử dụng hàm của PHP

https://www.vietjack.com/php/tong_hop_ham_trong_php.jsp

https://cafedev.vn/mien-phi-100-series-tu-hoc-php-tu-co-ban-toi-nang-cao/

Mảng cơ bản

❑Khai báo mảng

- Cú pháp: `$tên_mảng = array();`
- Ví dụ: khai báo mảng a
  - $a = array();

Mảng cơ bản

❑Khởi tạo mảng

- Cú pháp: `$tên_mảng = array([khóa=>] giá trị, ..., );`
- Các phần tử trong mảng cách nhau bằng dấu ","
- Với:
  - khóa: số nguyên dương/ chuỗi
  - Nếu khóa là chuỗi: dùng cặp nháy đôi "giá trị khóa" hoặc cặp nháy đơn 'giá trị khóa'
  - Mặc định, khóa tự động phát sinh, với phần tử đầu tiên của mảng khóa có giá trị là 0, phần tử thứ hai của mảng khóa có giá trị là 1, …

Mảng cơ bản

❑Truy xuất phần tử trong mảng

- Cú pháp: `$tên_mảng[<khóa>]`
- Ví dụ:

```
$mang1 = array(1, 5, 7);
$gia_tri_1 = $mang1[0]; //→ 1
$mang2 = array(1=>"Một", 2=>"Hai")
$gia_tri_3 = $mang[1]; //→ "Một
$mang2[3] = "Ba"; //Gán giá trị
```

Thao tác trên mảng

❑Đếm số phần tử trong mảng

- Cú pháp: `count($tên_mảng)`
- Ví dụ:

```
$mang = array(1, 2, 3, 4, 5, 6);
$so_phan_tu = count($mang); → 6
```

Thao tác trên mảng

❑ Duyệt mảng có khóa tự động

- Cú pháp:

```
for($i = 0; $i < $số_phần_tử; $i++)
{
  Xử lý các phần tử trong mảng (đọc, tính toán, thay đổi...)
  với mỗi phần tử: $tên_mảng[i]
}
```

- Ví dụ:

```
$mang = array(1, 2, 3, 4, 5, 6);
$n = count($mang);
for($i = 0; $i < $n; $i++)
    echo "\t" .$mang[$i];
→ 1  2  3  4  5  6
```

Thao tác trên mảng

❑ Duyệt mảng có khóa do người dùng tạo

- Duyệt để lấy và xử lý giá trị của các phần tử trong mảng:

```
foreach ($tên_mảng as $giá_trị)
{
  Xử lý các giá trị trong mảng (đọc, tính toán, thay đổi...)
  với mỗi phần tử: $giá_trị
}
```

- Ví dụ:

```
$mang = array(1=>"Một", 2=>"Hai", 3=>"Ba",
4=>"Bốn", 5=>"Năm");
foreach ($mang as $gia_tri)
    echo "\t $gia_tri";
→ Một Hai Ba Bốn Năm
```

Thao tác trên mảng

❑ Duyệt mảng có khóa do người dùng tạo

- Duyệt để lấy cả giá trị của khoá và giá trị của phần tử

```
foreach($tên_mang as $khóa=>$giá_trị)
{
  Xử lý
  với mỗi phần tử: $khoá=>$giá_trị
}
```

- Ví dụ:

```
$mang = array("mot"=>1, "hai"=>2, "ba"=>3,
"bon"=>4, "nam"=>5);
foreach ($mang as $khoa=>$gia_tri)
    echo "[$khoa] => $gia_tri, ";
→   [mot] => 1, [hai] => 2, [ba] => 3, [bon] =>
4   [nam] => 5
```

Thao tác trên mảng

❑ Tạo mảng từ chuỗi: Dùng explode()

- Chú ý: cần có quy ước cấu trúc cho chuỗi mảng
- Ví dụ: Quy ước khi nhập (tạo chuỗi mảng là các phần tử cách nhau bằng dấu ",")

```
$chuoi_mang = '1, 6, 3, 12, 8, 2';
 //tạo mảng
$mang = explode(',' , $chuoi_mang);
```

Thao tác trên mảng

❑ Xuất mảng ra chuỗi: Dùng implode()

- Chú ý: cần có quy ước cấu trúc cho chuỗi mảng
- Ví dụ: Quy ước khi xuất (xuất chuỗi mảng gồm các phần tử cách nhau bằng dấu ",")

```
$mang = array(1, 6, 3, 12, 8, 2);
//tạo chuỗi kết quả
$mang_chuoi = implode(',' , $mang);
echo $mang_chuoi;
```

Các hàm xử lý trên mảng

Vấn đề:

Cần xử lý mảng một cách nhanh chóng

Ví dụ:

· Tìm kiếm một giá trị trên mảng $mang

· Ghép mảng 1 với mảng 2 thành một mảng

Giải quyết:

Sử dụng hàm mảng trong thư viện hàm của PHP

```
$mang = array(0 => 'xanh', 1 => 'đỏ', 2 => 'tím', 3 => 'vàng');
// Tìm kiếm phần tử có giá trị "đỏ"
$khoa_do = array_search('đỏ', $mang); → 1, nếu không thấy trả về NULL
// Ghép mảng 1 với mảng 2 thành một mảng
$mang1 = array("màu" => "đỏ", 2, 4);
$mang2 = array("a", "b", "màu" =>"xanh", "hình" => "tròn", 4);
$mang_chung = array_merge($mang1, $mang2); // khi các mảng dùng để ghép có khóa trùng nhau thì mảng ghép sẽ chỉ lấy phần tử có khóa trùng của mảng cuối cùng:
Array ( [màu] => xanh, [0] => 2, [1] => 4, [2] => a, [3] => b, [hình] => tròn [4] => 4 )
```

Các hàm xử lý trên mảng

….

· Đếm số lần xuất hiện của các phần tử trong mảng $mang

· Tạo mảng duy nhất từ mảng hiện có

· Tìm các giá trị khác nhau của mảng 1 so với mảng 2

Giải quyết:

Sử dụng hàm mảng trong thư viện hàm của PHP

```
$mang = array(1, "hello", 1, "world", "hello", 2, "Chào", 1);
// Đếm số lần xuất hiện của các phần tử trong mảng
$mang_slxh = array_count_values($mang); // Array ([1] => 3; [hello] => 2; [world] => 1 ; [2] => 1 ; [Chào] => 1)
// Tạo mảng duy nhất từ mảng 1
$mang1 = array(1, 3, 1, 2, 5, 1, 3, 4);
$mang_duy_nhat= array_unique($mang1); -> 1, 3, 2, 5, 4
// Tìm giá trị khác nhau của mảng 1 so với mảng 2
$mang_1 = array("a"=>"xanh", "đỏ", "tím", "vàng");
$mang_2 = array("b"=>"xanh", "vàng","đỏ");
$mang_con_1 = array_diff($mang_1, $mang_2);
// Array ( [1] => tím )
```

LT hướng đối tượng (class) cơ bản

- Tạo lớp
  - Cú pháp:

```
class <tên lớp>
{ // khai báo các thuộc tính của lớp
  // gán và lấy giá trị của thuộc tính
  // các phương thức của lớp
}
```

LT hướng đối tượng (class) cơ bản

- Khai báo thuộc tính
  - Thuộc tính: thành phần lưu trữ các tính chất, đặc điểm của đối tượng.
  - Cú pháp

```
var <tên thuộc tính 1>;
…
```

  - Chú ý: Có thể thiết lập những giá trị mặc định ban đầu cho tất cả các thuộc tính được tạo ra từ lớp đó

```
class PHAN_SO
{
  var $tu_so = 1;
  var $mau_so = 1;
  ...
}
```

LT hướng đối tượng (class) cơ bản

- Xây dựng phương thức
  - Phương thức: Là chức năng mà đối tượng có thể thực hiện; có thể có hoặc không có giá trị trả về
  - Cú pháp:

```
function <tên phương thức> (các tham số truyền
  vào nếu có)
{
  // khối lệnh
}
```

LT hướng đối tượng (class) cơ bản

- Xây dựng phương thức
  - **Ghi chú**
    - Trong các phương thức của lớp, có thể truy cập các thuộc tính hay phương thức thông qua con trỏ $this
    - Con trỏ $this được dùng để chỉ lớp hiện tại đang làm việc.
    - Ví dụ: nếu một thuộc tính của lớp phân số có tên là $tu_so thì có thể tham chiếu đến nó như sau: $this->tu_so

LT hướng đối tượng (class) cơ bản

- Sử dụng lớp đối tượng
  - **Khởi tạo đối tượng**
    - Cú pháp: `$<tên biến đối tượng> = new <tên lớp>();`
    - Ví dụ:

```
$phan_so = new PHAN_SO();
```

LT hướng đối tượng (class) cơ bản

- Sử dụng lớp đối tượng
  - **Gán giá trị cho các thuộc tính của lớp**
    - Cú pháp: `$<tên biến đối tượng>-><tên thuộc tính> = <giá trị>;`
    - Ví dụ:

```
$phan_so->tu_so = 2;
$phan_so->mau_so = 4;
```

LT hướng đối tượng (class) cơ bản

- Sử dụng lớp đối tượng
  - ❑ **Gọi sử dụng phương thức của lớp**
    - **Phương thức không có giá trị trả về**
      - Cú pháp: `$<tên biến đối tượng>-><tên phương thức>(các giá trị truyền vào nếu có);`
      - Ví dụ:

```
$phan_so->khoi_tao_phan_so($ptu_so,$pmau_so);
```

LT hướng đối tượng (class) cơ bản

- Sử dụng lớp đối tượng
  - ❑ **Gọi sử dụng phương thức của lớp**
    - **Phương thức có giá trị trả về**
      - Cú pháp: `$<tên biến nhận giá trị> = $<tên biến đối tượng-> <tên phương thức>(các giá trị truyền vào nếu có);`
      - Ví dụ:

```
$phan_so_2 = New PHAN_SO();
$phan_so_2 = $phan_so->tong($a,$b);
```

LT hướng đối tượng (class) cơ bản

- Sử dụng lớp đối tượng
  - ❑ **Gọi sử dụng phương thức của lớp**
    - **Phương thức có giá trị trả về**
      - Cú pháp: `$<tên biến nhận giá trị> = $<tên biến đối tượng-> <tên phương thức>(các giá trị truyền vào nếu có);`
      - Ví dụ:

```
$phan_so_2 = New PHAN_SO();
$phan_so_2 = $phan_so->tong($a,$b);
```

Form và các điều khiển cơ sở

- FORM
  - ❑ **Thuộc tính action**
    - Quy định trang xử lý yêu cầu khi người dùng submit form.
    - Code HTML

```
<form action="vi_du_c2.php"
method="post" name="frm1">
```

Form và các điều khiển cơ sở

- Các điều khiển cơ sở
  - ❑ Đặc điểm chung
  - ❑ TextFiled/TextArea
  - ❑ Button
  - ❑ Checkbox
  - ❑ RadioButton/RadioGroup

Form và các điều khiển cơ sở

❑Đặc điểm chung

- Nằm trong thẻ Form
- Tên được thiết lập trong thuộc tính name của đối tượng.
- Có giá trị ban đầu (initial value) và giá trị hiện tại (current value), kiểu chuỗi.

Form và các điều khiển cơ sở

❑Đặc điểm chung

- Giá trị ban đầu không thay đổi, vì vậy khi reset Form, các giá trị hiện tại sẽ được đặt lại thành giá trị ban đầu.
- Có giá trị hiện tại (current value) khi Form submit
- Tạo một đối tượng: vào menu Insert => Form => chọn đối tượng muốn tạo. Hoặc cũng có thể tạo ra đối tượng bằng cách viết thẻ lệnh.

Form và các điều khiển cơ sở

❑TextField

- Vào Menu Insert => Form => chọn TextFiled
- Hoặc dùng thẻ input để tạo

```
<input type="text" name="textfield" value="" />
```

- Thiết lập các thuộc tính cơ bản như tên (name), độ rộng (size), số ký tự tối đa (maxlength)…

Form và các điều khiển cơ sở

❑TextArea

- Là TextField dạng multi line, dùng để nhập liệu trên nhiều dòng (multi line). Với:
  - Num lines: số dòng văn bản được hiển thị trên Textarea
  - Wrap: quy định việc hiển thị của văn bản có/không được phép tự động xuống dòng khi kích thước ngang của điều khiển không đủ để hiển thị nội dung văn bản. Mặc định là tự động xuống dòng.

Form và các điều khiển cơ sở

❑TextArea

- Vào Menu Insert => Form => chọn TextArea

- Hoặc dùng thẻ textarea để tạo:

```
<textarea name="textarea" cols=""
rows="3"> </textarea>
```

- Thiết lập các thuộc tính cơ bản như (name), độ rộng (cols), số dòng hiển thị (rows)…

Form và các điều khiển cơ sở

❑Button

- Có:
  - Submit button: khi nhấn button này thì thông tin sẽ postback về server. Trong một Form có thể có một hay nhiều Submit button.
  - Reset button: khi nhấn button này thì tất cả các đối tượng trên Form sẽ được reset trở lại giá trị ban đầu.

Form và các điều khiển cơ sở

❑Button

- Vào Menu Insert => Form => chọn Button

- Hoặc dùng thẻ input để tạo:

```
<input type="submit" name="Submit"
value="Submit" />
```

- Thiết lập các thuộc tính cơ bản như tên (name), và giá trị (value), kiểu (type)

Form và các điều khiển cơ sở

❑Checkbox

- Là đối tượng có hai trạng thái on/off (chọn/ không chọn). Nếu trạng thái checked được chọn thì Checkbox sẽ có giá trị là “checked”.
- Khi có nhiều Checkbox trong Form thì tại một thời điểm chúng ta có thể chọn một hay nhiều Checkbox (cũng có thể không chọn một Checkbox nào)

Form và các điều khiển cơ sở

❑Checkbox

- Vào Menu Insert => Form => chọn Checkbox
- Hoặc dùng thẻ input để tạo:

```
<input name="checkbox" type="checkbox"
 value="" checked />
```

- Thiết lập các thuộc tính cơ bản như tên (name), giá trị (value), trạng thái ban đầu.

Form và các điều khiển cơ sở

❑Checkbox

- Vào Menu Insert => Form => chọn Checkbox
- Hoặc dùng thẻ input để tạo:

```
<input name="checkbox" type="checkbox"
 value="" checked />
```

- Thiết lập các thuộc tính cơ bản như tên (name), giá trị (value), trạng thái ban đầu.

Form và các điều khiển cơ sở

❑RadioGroup

- Là một nhóm các RadioButton có cùng tên
- Khi một RadioButton đã được chọn (ở trạng thái on) thì tất cả các RadioButton cùng tên khác sẽ không được chọn (ở trạng thái off).

Form và các điều khiển cơ sở

RadioGroup

- Vào Menu Insert => Form => chọn RadioGroup
  - Thêm các RadioButton vào RadioGroup, đặt tên cho RadioGroup, thiết lập nội dung và giá trị cho các Radiobutton

Đọc giá trị từ điều khiển form

$_POST

- Được dùng để lấy giá trị của các điều khiển trên Form thông qua phương thức POST.
- Thông tin được gửi từ Form với phương thức POST không giới hạn lượng thông tin gửi đi và sẽ không được hiển thị trên địa chỉ URL nên người dùng không thể thấy được

Đọc giá trị từ điều khiển form

$_POST

- Cú pháp: lấy giá trị của một đối tượng trên Form sau khi Form submit:

```
$_POST["tên điều khiển"]
```

- Ví dụ: lấy giá trị TextField tên là txtTen

```
$ten = $_POST["txtTen"];
```

Thảo luận