ทฤษฎี บทที่ 5 เริ่มต้นกับ PHP

5.1 ทำความรู้จักกับ PHP

PHP เป็นภาษาสคริปต์แบบเซิร์ฟเวอร์ไซด์ (Server-side Script language) หมายถึง การประมวลผลจะเกิดขึ้นบนเครื่องแม่ข่ายหรือเซิร์ฟเวอร์(Server) แล้วจึงสร้างผลลัพธ์เป็นภาษา HTML ส่งให้กับเครื่องลูกข่ายหรือไคลเอ็นต์ (client) เพื่อแสดงผล ซึ่งลดภาระการส่งถ่ายข้อมูลจำนวนมากเพื่อมาประมวลผลบนเครื่องลูกข่าย

การเขียนสามารถทำโดยการเขียนโค้ด PHP แทรกลงไปในโค้ด HTML ด้วยการเปิดแท็ก <?php และปิดด้วยแท็ก ?> หรือเขียนเป็นโค้ด PHP อย่างเดียวก็ได้เช่นกัน และทำการบันทึกเป็นไฟล์ที่มีนามสกุล .php, php3 หรือ .phtml ขึ้นอยู่กับว่าได้กำหนดไว้ในการติดตั้งเว็บเซิร์ฟเวอร์หรือไม่

PHP จัดเป็นภาษาที่ง่ายในการเขียน สามารถนำมาใช้ทำเว็บเพจที่จำเป็นต้องมีการตอบสนองกับผู้ใช้ โดยเฉพาะอย่างยิ่ง PHP มีความสามารถในการนำข้อมูลจาก Database Server มาแสดงในเว็บเพจ จึงเหมาะแก่การนำมาใช้ทำเว็บบอร์ด, ไดนามิกเว็บเพจ, เว็บเมล์ เพื่อประโยชน์ในทางพาณิชย์อิเล็กทรอนิกส์ (e-commerce) ตลอดจนการสร้างเว็บแอพลิเคชันเพื่อใช้งานภายในองค์กรที่ต้องการคุณสมบัติในการเรียกใช้งานได้จากทุกที่

5.2 สิ่งที่จำเป็นต้องมีสำหรับการใช้ PHP

- เครื่องเซิร์ฟเวอร์ (จะใช้ PC ธรรมดาก็ได้)
- เครื่องไคลเอ็นต์ (เป็นเครื่องเดียวกับเครื่องเซิร์ฟเวอร์ก็ได้)
- โปรแกรมเว็บเซิร์ฟเวอร์ (Web Server) เช่น Apache, Microsoft Internet Information Server(IIS), Microsoft Web Server (PWS)
- โปรแกรม PHP
- โปรแกรมดาต้าเบสเซิร์ฟเวอร์ (Database Server) เช่น MySQL, PostgreSQL, Microsoft SQL Server (หากต้องมีการจัดเก็บข้อมูลในฐานข้อมูล)

5.3 รูปแบบของภาษา PHP

PHP จัดเป็นภาษาสคริปต์ภาษาหนึ่งที่ดำเนินการที่ฝั่งเซิร์ฟเวอร์ (server-side) คือเมื่อโค้ดถูกเรียกใช้โดย บราวเซอร์ โปรแกรม PHP ที่อยู่ในเครื่องที่เป็นเว็บเซิร์ฟเวอร์จะทำการประมวลผลแล้วสร้าง ผลลัพธ์ที่อยู่ในรูปของภาษา HTML ขึ้น แล้วจึงส่งมาให้กับเครื่องไคลเอ็นต์เพื่อให้บราวเซอร์แสดงผล ลักษณะการเขียนสคริปต์จะเขียนแทรกไว้ภายในไฟล์ HTML โดยเปิดด้วยแท็ก <?php และปิดด้วย ?>

การเขียน PHP แทรกใน HTML

```
1  <html>
2     <head>
3        <title>PHP First run</title>
4     </head>
5     <body>
6        <?php
7           print "My first PHP script.";
8        ?>
9     <br><br><font color=blue><h2>
10    Please enjoy the script.
11    </h2></font>
12    </body>
13 </html>
```

การเขียน PHP ล้วน ๆ

```
1  <?php
2     print "<title>PHP First run</title>";
3     print "<body>";
4     print "My first PHP script.";
5     print "<br><br><font color=blue><h2>";
6     print "Please enjoy the script.";
7     print "</h2></font>";
8     print "</body>";
9  ?>
```

5.4 <u>การใส่หมายเหตุ</u>

การใส่หมายเหตุ (Comment) ไว้ในสคริปต์ทำได้ 2 วิธี คือ

1. ใช้เครื่องหมาย // หรือ # ข้อความที่ตามหลังเครื่องหมายนี้จนถึงตัวอักษรสุดท้ายของบรรทัดนั้น ๆ จะไม่ถูกประมวลผล
2. ใช้เครื่องหมาย /* ร่วมกับ */ ข้อความที่อยู่ระหว่างเครื่องหมาย /* กับ */ จะไม่ถูกนำมาประมวลผล

5.5 <u>ตัวแปรและชนิดของตัวแปร (Variables and Types)</u>

โดยทั่วไปการใช้ตัวแปรใน PHP จะไม่จำเป็นต้องมีการระบุชนิด (type) เนื่องจากในขณะที่สคริปต์ถูกเรียกใช้งาน (runtime) PHP สามารถที่จะกำหนดให้เองว่าตัวแปรนั้น ๆ ควรเป็นตัวแปรชนิดใด โดยพิจารณาจากค่าที่ถูกกำหนดให้กับตัวแปรนั้น แต่หากต้องการกำหนดก็สามารถกำหนดชนิดของตัวแปรได้ในขณะที่สร้างตัวแปร (เรียกใช้ตัวแปรครั้งแรก) หรือหลังจากนั้นก็ได้

ตัวอย่างการกำหนดชนิดให้กับตัวแปร

ตัวอย่างที่ 1	*ผลลัพธ์*
$a_num = 12.535; print $a_num;	12.535

ตัวอย่างที่ 2	*ผลลัพธ์*
$a_num = (integer) 12.535; print $a_num;	12

ตัวอย่างที่ 3	*ผลลัพธ์*
$a_num = 12.535; settype($a_num, “interger”); print $a_num;	12

<u>การกำหนดชนิดของตัวแปรโดยวิธีตามตัวอย่างที่ 2 นั้นสามารถใช้คำสั่งดังต่อไปนี้</u>

(boolean) หรือ (boo)	เพื่อกำหนดให้เป็น Boolean
(interger) หรือ (int)	เพื่อกำหนดให้เป็น integer
(float) หรือ (double) หรือ (real)	เพื่อกำหนดให้เป็น float
(string)	เพื่อกำหนดให้เป็น string
(array)	เพื่อกำหนดให้เป็น array
(object)	เพื่อกำหนดให้เป็น object

<u>ส่วนการกำหนดโดยวิธีตามตัวอย่างที่ 3 โดยใช้ฟังก์ชั่น settype( ) มีรายละเอียด ดังนี้</u>

สามารถระบุค่าสตริงต่อไปนี้ลงในอาร์กิวเมนต์ตัวที่สอง

“boolean" “integer” “double” “string”

“array” “object”

5.6 <u>การตั้งชื่อตัวแปร</u>

การตั้งชื่อตัวแปรสามารถใช้ตัวอักษรภาษาอังกฤษ, ตัวเลข(0-9) และสัญลักษณ์ _(underscore) โดยจะต้องขึ้นด้วยตัวอักษร หรือ

สัญลักษณ์ _ เท่านั้น(จะนำหน้าด้วยตัวเลขไม่ได้) และจะต้องมีสัญลักษณ์ $ นำหน้า การเรียกใช้ตัวแปรนั้นจะต้องพิมพ์อักษรตัวใหญ่หรือเล็กให้ถูกต้อง มิฉะนั้นจะถูกมองเป็นตัวแปรคนละตัวกันหรือที่เรียกว่า case-sensitive ดังต่างเช่น $IDcard, $idcard และ $idCard จะถือว่าเป็นตัวแปรคนละตัวกัน

5.7 การตรวจสอบการมีอยู่ของตัวแปร และการยกเลิกตัวแปรด้วย isset( ), unset( )

เราสามารถตรวจสอบว่าตัวแปรได้ถูกสร้างขึ้นหรือไม่ด้วยฟังก์ชั่น isset( ) หากเป็นตัวแปรที่ถูกสร้างขึ้นแล้วจะให้ค่าเป็นจริง(true) มิฉะนั้นจะให้ค่าเป็นเท็จ (false)

ตัวอย่างที่ 5.1 *การตรวจสอบการมีอยู่ของตัวแปรด้วยฟังก์ชั่น isset( )*

```php
<?php
   $a_num = 12.535;
   if(isset($a_num)){
      print("Value of \$a_num is $a_num.");
   } else {
      print("The variable does not exists.");
   }
?>
```

ผลลัพธ์

Value of $a_num is 12.535.

แต่หากลบบรรทัดแรกที่เป็นการกำหนดค่าให้กับตัวแปร $a_num ออก หรือใส่เครื่องหมาย // ไว้หน้าบรรทัดนี้ ผลลัพธ์ที่ได้จะเป็นดังนี้

The variable does not exists.

และสำหรับตัวแปรที่ถูกสร้างขึ้นแล้ว หากต้องการกำจัดทิ้ง สามารถทำได้โดยใช้ฟังก์ชั่น unset( )

5.8 ขอบเขตของตัวแปร (Variable Scope)

ตัวแปรที่ถูกสร้างขึ้น ปกติแล้วจะเป็นตัวแปรแบบ Local คือ ตัวแปรที่ถูกสร้างขึ้นภายในฟังก์ชั่นก็จะเรียกใช้ไม่ได้จากนอกฟังก์ชั่น และตัวแปรที่ถูกสร้างขึ้นนอกฟังก์ชั่น ก็จะเรียกใช้ไม่ได้จากในฟังก์ชั่นเช่นกัน เว้นแต่ว่าจะสร้างตัวแปรภายในฟังก์ชั่นให้เป็นแบบ Global

ตัวอย่างที่ 5.2 *ตัวแปรแบบ Local ที่อยู่นอกฟังก์ชั่น ไม่สามารถถูกเรียกใช้จากภายในฟังก์ชั่นได้*

```php
<?php
   function ShowProvince( ){
      echo "My province : $myprovince";
   }
   $myprovince = "Nakornrachasima";
   ShowProvince( );
?>
```

ผลลัพธ์

My province :

ตัวอย่างที่ 5.3 *การเรียกใช้ตัวแปรแบบ Local ของฟังก์ชั่นจากภายในฟังก์ชั่นเอง*

```php
<?php
   function ShowProvince( ){
      $myprovince = "Nakornrachasima";
      echo "My province : $myprovince";
   }
   ShowProvince( );
?>
```

ผลลัพธ์

My province : Nakornrachasima

ตัวอย่างที่ 5.4	*การเรียกใช้ตัวแปรแบบ Local ของฟังก์ชั่นจากภายในฟังก์ชั่นเอง*

```php
<?php
   function ShowProvince( ){
      global $myprovince ;
      echo "My province : $myprovince";
   }
   $myprovince = "Nakornrachasima";
   ShowProvince( );
?>
```

ผลลัพธ์

My province : Nakornrachasima

ในตัวอย่างที่ 5.4 ตัวแปร $myprovince ถูกกำหนดให้เป็นตัวแปรแบบ global จึงสามารถรับค่าจากภายนอกฟังก์ชั่นมาแสดงได้

ชนิดของข้อมูลและตัวแปรใน PHP มี 8 ชนิด ดังต่อไปนี้

boolean	integer	float	string	array
object	resource	NULL		

5.9 <u>โอเปอเรเตอร์ (Operator)</u>

โอเปอเรเตอร์เกี่ยวกับตัวเลข (Arithmetic Operator)

โอเปอร์เรเตอร์	ชื่อ	ตัวอย่าง	ความหมาย
+	Addition	$a + $b	การบวกค่า $a กับ $b
-	Subtraction	$a - $b	การลบค่า $b ออกจากค่า $a
*	Multiplication	$a * $b	การคูณค่า $a กับ $b
/	Division	$a / $b	การหารค่า $a ด้วย $b
%	Modulation	$a % $b	การหาค่าเศษที่เหลือจากการหารค่า $a ด้วย $b

โอเปอเรเตอร์สำหรับกำหนดค่า (Assignment Operators)

โอเปอร์เรเตอร์	ตัวอย่าง	ความหมาย
=	$a = $b	การกำหนดค่าให้ตัวแปรทางซ้ายมือมีค่าเท่ากับค่า หรือตัวแปรทางขวามือ
+=	$a += $b	การกำหนดค่าให้ตัวแปรทางซ้ายมือเพิ่มค่าขึ้นเท่ากับค่า หรือตัวแปรทางขวามือ
-=	$a -= $b	การกำหนดค่าให้ตัวแปรทางซ้ายมือลดค่าลงเท่ากับค่าหรือตัวแปรทางซ้ายมือ
.=	$a .= $b	การกำหนดค่าให้ตัวแปรสตริงทางซ้ายมือถูกต่อท้ายด้วยข้อความทางขวามือ

ตัวอย่างที่ 5.5	*การใช้โอเปอเรเตอร์ =, +=, -=*

```php
<?php
   $a = $b = 15;
   $a += 5;  $b -= 5;
   print "\$a = " . $a . "<br>\n";
   print "\$b = " . $b . "<br>\n";
?>
```

ผลลัพธ์

$a = 20
$b = 10

ตัวอย่างที่ 5.6	*การใช้โอเปอรเรเตอร์ .=*

```php
<?php
   $a = "สุขสันต์วันเกิด คุณ ";
   $name = "John";
   $a .= $name;
   print $a . "\n";
?>
```

ผลลัพธ์

สุขสันต์วันเกิด คุณ John

การกำหนดค่าดังที่แสดงในตัวอย่างที่ 5.5 และ 5.6เป็นการกำหนดด้วยค่า(assign by value) ซึ่งเป็นการสำเนาค่าให้แก่ตัวแปรทางซ้ายมือ ดังนั้น หากภายหลังตัวแปรดั้งเดิมทางขวามือมีค่าเปลี่ยนไป จะไม่ทำให้ค่าของตัวแปรทางซ้ายมือเปลี่ยนแปลงไปด้วย

ยังมีวิธีการกำหนดค่าแบบกำหนดด้วยการอ้างอิง (assign by reference) โดยใส่เครื่องหมาย & หน้าตัวแปรทางขวามือ ซึ่งเป็นการกำหนดให้ตัวแปรทางซ้ายมือชี้ไปที่ข้อมูลเดียวกับตัวแปรทางขวามือ ดังนั้นเมื่อค่าของตัวแปรทางขวามือเปลี่ยนไป ตัวแปรทางซ้ายก็จะมีค่าเปลี่ยนไปด้วย

ตัวอย่างที่ 5.7	*การกำหนดค่าด้วยการอ้างอิง แสดงให้เห็นว่าค่าของ $a จะเปลี่ยนตามค่าของ $name*

```php
<?php
   $name = "John";
   $a = &$name;
   print $a . "<br>";
   $name = "Bill";
   print $a . "<br>";
?>
```

ผลลัพธ์

John
Bill

โอเปอเรเตอร์การเปรียบเทียบ (Comparison Operators)

โอเปอร์เรเตอร์	ชื่อ	ตัวอย่าง	ความหมาย
= =	Equal	$a = = $b	เป็นจริงเมื่อ $a เท่ากับ $b
= = =	Identical	$a = = = $b	เป็นจริงเมื่อ $a เท่ากับ $b และชนิดข้อมูลเหมือนกัน
!=	Not equal	$a != $b	เป็นจริง เมื่อ $a ไม่เท่ากับ $b
< >	Not equal	$a <> $b	เป็นจริงเมื่อ $a ไม่เท่ากับ $b
!= =	Not identical	$a != = $b	เป็นจริงเมื่อ $a ไม่เท่ากับ $b หรือเท่ากันแต่ชนิดข้อมูลต่างกัน
<	Less than	$a < $b	เป็นจริงเมื่อ $a น้อยกว่า $b
>	Greater than	$a > $b	เป็นจริงเมื่อ $a มากกว่า $b
<=	Less than or equal to	$a <= $b	เป็นจริงเมื่อ $a น้อยกว่าหรือเท่า $b
>=	Greater than or equal to	$a >= $b	เป็นจริงเมื่อ $a มากกว่าหรือเท่ากับ $b

ตัวอย่างที่ 5.8	*แสดงความแตกต่างระหว่างโอเปอเรเตอร์ == และ ===*

```php
<?php
   $a = 15;  $b = "15";
   print "\$a is " . gettype($a) . " = $a<br>";
   print "\$b is " . gettype($b) . " = $b<br>";
   if ($a == $b) {
      print "\$a == \$b<br>";
   } else {
      print "\$a != \$b<br>";
   }
   if ($a ===$b) {
      print "\$a === \$b<br>";
   } else {
      print "\$a !== \$b<br>";
   }
?>
```

ผลลัพธ์

```
$a is integer = 15
$b is string = 15
$a == $b
$a !== $b
```

จากตัวอย่างที่ 5.8 จะเห็นได้ว่าตัวแปร $a ถูกกำหนดค่าเป็น 15 จึงมีชนิดข้อมูลเป็น integer ส่วนตัวแปร $b ถูกกำหนดให้เป็น “15” จึงมีชนิดข้อมูลเป็น string ดังนั้นเมื่อนำมาเปรียบเทียบกันด้วยโอเปอเรเตอร์ = = จึงให้ผลลัพธ์เป็น true เนื่องจากไม่มีการเปรียบเทียบชนิดของข้อมูลด้วย แต่เมื่อเปรียบเทียบด้วยโอเปอเรเตอร์ = = = จะให้ผลลัพธ์เป็น false เพราะตัวแปรทั้งสองมีชนิดของข้อมูลต่างกัน

โอเปอเรเตอร์ควบคุมข้อผิดพลาด (Error Control Operator)

โอเปอรเรเตอร์สำหรับควบคุมข้อผิดพลาดมีเพียงตัวเดียวคือ @ ซึ่งเมื่อให้นำหน้า expression ใดๆ แล้วจะทำให้ไม่มีการแสดงข้อผิดพลาดออกมาหากมีข้อผิดพลาดเกิดขึ้น

ตัวอย่างที่ 5.9	เปรียบเทียบผลลัพธ์เมื่อมีการใช้และไม่ใช้โอเปอเรเตอร์ @

```php
<?php
   $filename = "MyProject.doc";
   $file_h = @fopen($filename, "r");
   if($file_h) {
      print "พบไฟล์ $filename";
   } else {
      print "ไม่พบไฟล์ $filename";
   }
?>
```

ผลลัพธ์

ไม่พบไฟล์ MyProject.doc

แต่หากไม่มีการใช้โอเปอเรเตอร์ @ จะให้ผลลัพธ์ดังนี้

Warning: fopen(MyProject.doc): failed to open stream: No such file or directory in **c:\appserv\www\lesson5_2550\ex5_9.php** on line **3**
พบ MyProject.doc

โอเปอเรเตอร์การเพิ่ม / ลดค่า (Increment / Decrementing Operators)

โอเปอร์เรเตอร์	ชื่อ	ความหมาย
++$a	Pre-increment	เพิ่มค่าตัวแปร $a ขึ้น 1 แล้วจึงคืนค่าตัวแปร $a
$a++	Post-increment	คืนค่าตัวแปร $a แล้วจึงเพิ่มค่าตัวแปร $a ขึ้นอีก 1
- -$a	Pre-decrement	ลดค่าตัวแปร $a ลง 1 แล้วจึงคืนค่าตัวแปร $a
$a- -	Post-decrement	คืนค่าตัวแปร $a แล้วจึงลดค่าตัวแปร $a ลง 1

ตัวอย่างที่ 5.10	แสดงการใช้โอเปอเรเตอร์ทั้งสี่แบบ

```php
<?php
   $a = 100;
   print "\$a = $a<br>";
   print "++\$a = " . ++$a . "<br>";
   print "Print \$a again = $a<br><br>";
   $b = 100;
   print "\$b = $b<br>";
   print "\$b++ = " . $b++ . "<br>";
   print "Print \$b again = $b<br><br>";
   $c = 100;
   print "\$c = $c<br>";
   print "- -\$c = ". —$c . "<br>";
   print "Print \$c again = $c<br><br>";
   $d = 100;
   print "\$d = $d<br>";
   print "\$d- - = " . $d— . "<br>";
   print "Print \$d again = $d<br><br>";
?>
```

ผลลัพธ์

```
$a = 100
++$a = 101
Print $a again = 101

$b = 100
$b++ = 100
Print $b again = 101

$c = 100
- -$c = 99
Print $c again = 99

$d = 100
$d- - = 100
Print $d again = 99
```

โอเปอเรเตอร์ทางตรรกะ (Logical Operators)

โอเปอร์เรเตอร์	ชื่อ	ตัวอย่าง	ความหมาย
and หรือ &&	And	$a and $b	จะให้ค่าเป็นจริงเมื่อ $a และ $b เป็นจริงทั้งคู่ กรณีที่เหลือจะให้ค่าเป็นเท็จ
or หรือ ||	Or	$a or $b	จะให้ค่าเป็นเท็จเมื่อ $a หรือ $b เป็นเท็จทั้งคู่ กรณีที่เหลือจะให้ค่าเป็นจริง
xor	Exclusive Or	$a xor $b	จะให้ค่าเป็นเท็จเมื่อ $a และ $b เป็นจริงทั้งคู่หรือเป็นเท็จทั้งคู่ กรณีที่เหลือจะให้ค่าเป็นจริง
!	Not	!$a	จะให้ค่าเป็นจริงเมื่อ $a เป็นเท็จ และจะให้ค่าเป็นเท็จเมื่อ $a เป็นจริง

โอเปอเรเตอร์สตริง (String Operators)

โอเปอเรเตอร์สตริงมีอยู่ 2 ตัวคือ . และ .= โดยโอเปอเรเตอร์ . มีหน้าที่เชื่อมข้อความสอบข้อความเข้าเป็นข้อความเดียวกัน ส่วน .= ใช้สำหรับกำหนดค่าให้นำข้อความอื่นมาเชื่อมต่อกับข้อความเดิม

ตัวอย่างที่ 5.11	*การใช้โอเปอเรเตอร์ . เพื่อเชื่อมข้อความในตัวแปร $a และ $name เข้าด้วยกันและเก็บไว้ใน $a*

```php
<?php
   $a = "สุขสันวันเกิดคุณ ";
   $name = "John";
   $a = $a . $name;
   print $a . "\n";
?>
```

ผลลัพธ์

```
สุขสันวันเกิดคุณ John
```

โอเปอเรเตอร์เกี่ยวกับอาร์เรย์ (Array Operator)

โอเปอเรเตอร์เกี่ยวกับอาร์เรย์มีอยู่เพียงตัวเดียวคือ + ซึ่งใช้เชื่อมค่าอาร์เรย์ 2 ชุดเข้าด้วยกัน โดยในกรณีที่มีคีย์ซ้ำกันจะไม่มีการซ้อนทับข้อมูลเดิม

ตัวอย่างที่ 5.12	*การเชื่อมค่าอาร์เรย์ในตัวแปร $executive และ $staffer เข้าด้วยกันโดยโอเปอเรเตอร์ +*

```
1  <?php
2     $executive = array(
3        "Somboon" => "Managing Director",
4        "Eakachai" => "Accounting Manager",
5        "Sunya" => "Production Manager");
6     $staffer = array(
7        "Eakachai" => "System analysis",
8        "Pongsak" => "Programmer",
9        "Sunya" => "Scientist",
10       "Sansern" => "Salesman");
11    $all = $executive + $staffer;
12    foreach ($all as $name => $position) {
13       print "$name is $position<br>";
14    }
15 ?>
```

ผลลัพธ์

```
Somboon is Managing Director
Eakachai is Accounting Manager
Sunya is Production Manager
Pongsak is Programmer
Sansern is Salesman
```

จากตัวอย่าง จะเห็นว่าเมื่อค่าอาร์เรย์ในตัวแปร $staffer ถูกเชื่อมเข้ากับค่าอาร์เรย์ในตัวแปร $executive แล้ว ค่าในอาร์เรย์ทางขวามือ ($staffer) ที่มีคีย์ซ้ำกับในอาร์เรย์ทางซ้ายมือ ($executive) จะไม่ปรากฏในอาร์เรย์ผลลัพธ์ (คือ “Eakachai” => “System analysis” และ “Sunya” => “Scientist”)

5.10 <u>ประโยคควบคุม (Control Statements)</u>

เนื่องจาก PHP ได้ยืมรูปแบบโครงสร้างจากภาษา C ประโยคควบคุมส่วนใหญ่จึงไม่ต่างจากภาษาซีมากนัก ดังตารางต่อไปนี้

if( )	else	elseif( )	endif	switch( )
while( )	do…while( )	for( )	foreach( )	break
continue	declare( )	return( )	require( )	include( )
require_once( )	include_once( )			

คำสั่ง if…else, if…elseif

เป็นคำสั่งที่ใช้เปรียบเทียบเงื่อนไขเพื่อเลือกดำเนินการอย่างใดอย่างหนึ่ง เช่นเดียวกับที่ใช้ในภาษาอื่น ๆ โครงสร้างที่ใช้จะขึ้นกับวัตถุประสงค์ บางกรณีสามารถเขียนได้หลายลักษณะ ดังตัวอย่างต่อไปนี้

<u>ตัวอย่างที่ 1</u>	*<u>ผลลัพธ์ เมื่อตัวแปร $a เท่ากับ $z</u>*

if ($a == $z) {
print “The value is ” . $z;
}
	The value is ค่าของตัวแปร $z

<u>ตัวอย่างที่ 2</u>	*<u>ผลลัพธ์เมื่อตัวแปร $a ไม่เท่ากับ $z</u>*

if ($a == $z) {
print “The value is ” . $z;
} else {
print “The value is not ” . $z;
}
	The value is not ค่าของตัวแปร $z

ตัวอย่างที่ 3	*ผลลัพธ์ถ้าไม่ใช่ทั้งสอบกรณีแรก*

if ($a == $z) {
print "The value is " . $z;
} elseif ($a == $y) {
print "The value is not " . $y;
} else {
print "Invalid value.";
}
	The value is not ค่าของตัวแปร $z

คำสั่ง switch และคำสั่ง break

ในกรณีที่ต้องการเปรียบเทียบค่าเพื่อหาทางเลือกจากหลาย ๆ ทางเลือกว่าจะให้ดำเนินการตามทางเลือกใด การใช้ switch อาจช่วยให้เกิดความสับสนน้อยกว่าการใช้ if และสามารถให้ผลลัพธ์แบบเดียวกัน ตัวอย่างเช่น

ตัวอย่างที่ 5.13 *การใช้คำสั่ง switch…case*

```
1  <?php
2     $choice = 7;
3     switch($choice) {
4        case 1 :  print "Your choice is ONE <br>";
5           break;
6        case 2 :  print "Your choice is TWO <br>";
7           break;
8        default:  print "Another<br>";
9     }
10 ?>
```

ผลลัพธ์

Another

คำสั่ง while, do…while

คำสั่ง while ใช้เมื่อต้องการให้มีการดำเนินการเป็นวงรอบอย่างมีเงื่อนไข คือ ให้ดำเนินการประโยคคำสั่งภายใน while ไปเรื่อย ๆ จนกว่าเงื่อนไขของ while จะเป็นเท็จ

ตัวอย่างที่ 5.14 *การทำงานเป็นวงรอบด้วยประโยค while*

```
1  <?php
2     $j = 1;  $k = 1;
3     while ($j <= 10){
4        print $j . "<br>";
5        $j += $k;
6        $k++;
7     }
8  ?>
```

ผลลัพธ์

1
2
4
7

ส่วนการทำงานของ do…while จะคล้ายกับ while แต่ในตอนเริ่มต้นลูปจะไม่มีการตรวจสอบเงื่อนไข ดังนั้นจึงมีการดำเนินการอย่างน้อยหนึ่งวงรอบเสมอ ผิดกับกรณีของ while ซึ่งจะมีการตรวจสอบเงื่อนไขก่อนการดำเนินการคำสั่งภายในลูป ซึ่งหากเงื่อนไขเป็นเท็จก็จะไม่ดำเนินการใด ๆ ภายในลูปเลย

ตัวอย่างที่ 5.15	*การทำงานเป็นวงรอบด้วยประโยค do...while*

```php
<?php
   $j = 101;  $k = 1;
   do {
      print $j . "<br>";
      $j += $k;
      $k++;
   } while ($j <= 100)
?>
```

ผลลัพธ์

101

คำสั่ง for

รูปแบบการใช้คำสั่ง for เป็นดังนี้

```
for (expression1 ; expression2 ; expression3) {
        statement;
}
```

expression1	ส่วนของการกำหนดค่าเริ่มต้นให้กับตัวแปร
expression2	ส่วนของการกำหนดเงื่อนไขเพื่อใช้เปรียบเทียบ
expression3	ส่วนของการกำหนดให้มีการเพิ่มหรือลดค่าของตัวแปรในแต่ละรอบที่ดำเนินการ

ตัวอย่างที่ 5.16	*การทำงานเป็นวงรอบด้วยประโยค for*

```php
<?php
   for($i = 1; $i <= 10; $i++){
      print $i . "<br>";
      if($i >= 5 ) {
         break;
      }
   }
?>
```

ผลลัพธ์

1
2
3
4
5

คำสั่ง foreach

ตั้งแต่ PHP 4 เป็นต้นมา เรามีทางเลือกในการทำงานกับตัวแปรอาร์เรย์มากขึ้น โดยสามารถใช้ประโยค foreach ดึงข้อมูลและคีย์ในตัวแปรอาร์เรย์ออกมาทำงานเป็นวงรอบได้โดยไม่ต้องใช้ลูป while คำสั่ง foreach สามารถใช้ได้ 2 รูปแบบ คือ

รูปแบบที่ 1

```
foreach(array_expression as $value) {
        statement;
}
```

ตัวอย่างที่ 5.17	*การใช้ foreach แบบที่ 1*

```php
<?php
   $arr_name = array("aaa","bbb","ccc","ddd","fff","ggg");
   foreach ($arr_name as $each_name) {
      print "$each_name  ";
   }
?>
```

ผลลัพธ์

aaa bbb ccc ddd fff ggg

ประโยคแรกเป็นการเก็บรายชื่อเข้าไว้ในตัวแปรอาร์เรย์ $arr_name ส่วนในประโยคที่สอง foreach จะทำให้เกิดการทำงานเป็นวงรอบ โดยแต่วงรอบ ค่าของตัวแปรอาร์เรย์ $arr_name แต่ละค่าตั้งแต่ค่าแรกเป็นต้นไปจะถูกใส่ไว้ในตัวแปร $each_name แล้วจึงดำเนินการตามประโยคถัดไปภายในลูป คือแสดงค่าแต่ละค่าของตัวแปรอาร์เรย์ $arr_name ให้ปรากฏบนจอภาพ

รูปแบบที่ 2

```
foreach(array_expression as $key => $value) {
        statement;
}
```

รูปแบบนี้นอกจากจะมีการส่งค่าให้กับตัวแปร $value แล้ว ยังมีการส่งค่าคีย์ให้กับตัวแปร $key อีกด้วย

ในการกำหนดค่าให้กับตัวแปรอาร์เรย์ หากเราไม่ระบุคำที่ใช้เป็นคีย์ โปรแกรมจะกำหนดให้เป็นเลขจำนวนเต็มตั้งแต่ 0 เป็นต้นไปตามลำดับ เช่น

$arr_name = array("A", "B", "C", "D", "E");

ตัวอย่างที่ 5.18 *การใช้ foreach แบบที่ 2*

```
1 <?php
2   $City = array("Australia"=>"Canberra", "Brazil"=>"Brasilia",
3                 "Canada"=>"Ottawa");
4   foreach ($City  as  $Country =>  $eachCity) {
5         print  "The capital of  $Country id  $eachCity<br>";
6   }
7 ?>
```

ผลลัพธ์

```
The capital of Australia id Canberra
The capital of Brazil id Brasilia
The capital of Canada id Ottawa
```

การทำงานของประโยค foreach ข้างบน นอกจากจะมีการนำชื่อเมืองหลวง (เป็นข้อมูล)ไปใส่ไว้ในตัวแปร $eachCity แล้ว ยังนำชื่อประเทศ(เป็นคีย์) ไปใส่ไว้ในตัวแปร $Country ในแต่ละวงรอบอีกด้วย

คำสั่ง require, include

เราจะใช้คำสั่ง require และ include เมื่อต้องการแทรกสคริปต์จากไฟล์หนึ่งเข้าไปในอีกไฟล์หนึ่ง โดยจะให้ผลลัพธ์เหมือนกับว่าสคริปต์ที่แทรกเข้ามาถูกพิมพ์ไว้ในไฟล์นั้นโดยตรง จึงเหมาะในการแยกการกำหนดค่าตัวแปร, ค่าคงที่ และรูทีนหรือฟังก์ชั่นการทำงานที่ใช้บ่อยไว้ในไฟล์ๆ หนึ่ง เพื่อเอาไว้ include เข้าไปในไฟล์อื่น ดังตัวอย่างนี้

ไฟล์ test.inc

```
1 <?php
2       $Lang = "Thai";
3       $Country = "Thailand";
4       function View( ){
5             global  $Lang, $Country;
6             print  "My language is $Lang. <br>";
7             print "My country is $Country. <br>";
8       }
9 ?>
```

ตัวอย่างที่ 5.19	*ใช้คำสั่ง include เพื่อแทรกสคริปต์จากไฟล์ test.inc เข้ามาในไฟล์ ex5_19.php*

```
1 <?php
2   include("test.inc");
3   $Lang = "English";
4   View( );
5 ?>
```

ผลลัพธ์

My language is English.
My country is Thailand.

ในตัวอย่างที่ 5.19 นี้จะใช้ require แทน include ก็ได้ แต่สิ่งที่ต่างกันระหว่าง require และ include ก็คือ ในกรณีที่เกิดข้อผิดพลาดจากการทำงานของคำสั่ง เช่นหาไฟล์ที่ต้องนำมาแทรกไม่พบ คำสั่ง include จะแสดงข้อผิดพลาดเป็น “คำเตือน” (Warning) และจะยังคงทำงานต่อไป ในขณะที่คำสั่ง require จะแสดงข้อผิดพลาดเป็น Fatal Error (ข้อผิดพลาดร้ายแรง) และหยุดการทำงานของเพจนั้นทันที

ตัวอย่างที่ 5.20	*แสดงข้อผิดพลาดจากคำสั่ง include เนื่องจากหาไฟล์ไม่พบ*

```
1 <?php
2   include("file.inc");
3   print "OK";
4 ?>
```

ผลลัพธ์

Warning: main(): Failed opening 'file.inc' for inclusion (include_path='.;c:\php4\pear') in **c:\appserv\www\lesson5_2550\ex5_20.php** on line **2**
OK

ตัวอย่างที่ 5.21	*แสดงข้อผิดพลาดจากคำสั่ง require เนื่องจากหาไฟล์ไม่พบ*

```
1 <?php
2   require("file.inc");
3   print "OK";
4 ?>
```

ผลลัพธ์

Fatal error: main(): Failed opening required 'file.inc' (include_path='.;c:\php4\pear') in **c:\appserv\www\lesson5_2550\ex5_21.php** on line **2**

คำสั่ง require_once, include_once

คำสั่ง require_once และ include_once ใช้เมื่อต้องการแทรกไฟล์สคริปต์หนึ่งเข้าในอีกไฟล์หนึ่งสคริปต์หนึ่งเช่นเดียวกับ require และ include แต่ require_once และ include_once จะเป็นการป้องกันไม่ให้มีการแทรกไฟล์ซ้ำในกรณีที่ไฟล์สคริปต์นั้นเคยถูกเรียกมาแล้วครั้งหนึ่ง เพื่อเป็นการป้องกันการประกาศฟังก์ชั่นซ้ำ หรือการกำหนดค่าให้ตัวแปรซ้ำ ซึ่งจะทำให้เกิดข้อผิดพลาดขึ้น

ตัวอย่างที่ 5.22	*แสดงข้อผิดพลาดจากการ include ซ้ำ*

```
1 <?php
2   include("test.inc");
3   $Lang = "English";
4   include("test.inc");
5   View( );
6 ?>
```

ผลลัพธ์

Fatal error: Cannot redeclare view() (previously declared in c:\appserv\www\lesson5_2550\test.inc:5) in **c:\appserv\www\lesson5_2550\test.inc** on line **4**

ในกรณีที่มีการสร้างสคริปต์ย่อยเพื่อเก็บค่าตัวแปรหรือฟังก์ชั่นที่ต้องมีการเรียกใช้บ่อย ๆ ในไฟล์สคริปต์อื่น ๆ หากขาดความระมัดระวังอาจเกิดปัญหาการใช้ include หรือ require ซ้ำดังที่ได้แสดงในตัวอย่างที่ 5.22 เนื่องจากเราอาจสร้างสคริปต์ย่อยเพื่อเรียกกันเองได้หลายชั้น จนเกิดความสับสนได้ว่าไฟล์สคริปต์ใดได้ถูกแทรกเข้ามาในสคริปต์หลักแล้วบ้าง