Pertemuan 12

Array dalam PHP

Array merupakan sekumpulan elemen yang memiliki tipe sama. Dalam array data tersimpan dengan menggunakan index untuk memudahkan pencarian kembali data tersebut. Data yang berada dalam sebuah array disebut juga dengan elemen – elemen array. Semua elemen array mempunyai tipe data yang sama. Aray dapat memiliki dimensi lebih dari satu (*multidimensi*). Array berguna dalam suatu pemrograman yang memerlukan beberapa variabel yang akan menampung data dengan tipe data yang sama dan akan mendapat perlakuan yang serupa.
Indeks pada array dapat berupa angka atau huruf (*string*). Jika indeks array berbentuk angka, maka array tersebut akan disebut sebagai **indexed array** atau **vektor array**. Sedangkan bila indeks array berbentuk string maka array tersebut akan disebut sebagai **associative array**.

Aturan penulisan array dalam PHP adalah sebagai berikut :

$nama_array[no_indeks] atau **$nama_array[“str_indeks”]**

$nama_array adalah nama variabel yang digunakan sebagai nama array. **no_indeks** adalah nomor indeks dari vector array yang dimulai dari **0**. Sedangkan **str_indeks** adalah string yang digunakan sebagai indeks untuk associative array.

Contoh array sederhana :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$nama = array('Anita','Shinta','Tiara','Yuanita');

echo $nama[2];

?>

</body>
</html>
```

Contoh vektor array :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$kota[] = "Jakarta";
$kota[] = "Bandung";
$kota[] = "Surabaya";
$kota[] = "Yogyakarta";
$kota[] = "Semarang";

echo $kota[1];
echo '<br>';
print ("$kota[3]");

?>

</body>
</html>
```

Contoh associative array :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$merk["satu"] = "Nissan";
$merk["dua"] = "Honda";
$merk["tiga"] = "Mazda";
$merk["empat"] = "Toyota";
$merk["lima"] = "Daihatsu";

echo "Saya sedang menggunakan mobil : ";
echo $merk["dua"];

?>

</body>
</html>
```

Perulangan pada PHP

Perulangan atau *looping* pada PHP digunakan untuk menjalankan sebuah operasi secara berulang – ulang tanpa harus menuliskan kode operasi tersebut secara berulang. Hal ini akan membuat kode yang dibuat menjadi lebih ringkas dan rapi.

1. WHILE

   Perulangan jenis ini akan melakukan *looping* hingga persyaratan / kondisi yang diberikan pada perintah **while** terpenuhi. Bentuk umum dari perulangan ini adalah :

```
while(kondisi)
{
  pernyataan
}
```

Dalam bentuk perulangan WHILE, sepanjang kondisi bernilai true (benar), maka seluruh pernyataan yang berada di dalam kurung kurawal akan dijalankan secara berulang dan akan berkakhir jika kondisi (telah) bernilai false (salah).

Contoh perulangan WHILE sederhana:

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$a = 1;

while ($a <= 100)
  {
      echo $a;
      echo "<br>";
      $a++;
  }

?>

</body>
</html>
```

2. DO – WHILE

Cara kerja perulangan do – while hampir sama dengan perulangan while, hanya saja pada perulangan do – while sebuah pernyataan paling sedikit (*minimal*) berjalan 1 (*satu*) kali. Hal ini berbeda dengan perulangan while yang tidak akan menjalankan penyataan jika tidak memenuhi kondisi yang diberikan. Bentuk umum dari perulangan ini adalah sebagai berikut :

```
do
{
  pernyataan;
}
while(kondisi);
```

Contoh perulangan DO – WHILE sederhana :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$a = 1;

do
{
  print ("Belajar PHP ke : <b> $a </b>");
  print ("<br>");
  $a++;
}

while($a <= 10);

?>

</body>
</html>
```

3. FOR

Beda dengan dua perulangan sebelumnya, perulangan FOR digunakan untuk melakukan perulangan yang jumlah perulangannya telah ditentukan sebelumnya. Bentuk umum dari perulangan ini adalah sebagai berikut :

```
for(ekspresi_1; ekspresi_2; ekspresi_3;)
{
  pernyataan_pernyataan;
}
```

Pada bentuk di atas dapat dilihat terdapat 3 (*tiga*) macam bentuk ekspresi yang masing – masing berfungsi sebagai berikut :

- **ekspresi_1** berfungsi untuk memberikan nilai awal variabel yang akan digunakan untuk melakukan pengulangan.
- **ekspresi_2** berfungsi sebagai syarat atau kondisi yang harus terpenuhi agar pengulangan dapat belangsung (*bernilai true*).
- **ekspresi_3** berfungsi untuk mengatur nilai variabel yang digunakan dalam **ekspresi_1**.

Contoh perulangan FOR sederhana :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body bgcolor = "black">
<font color="white" face = "calibri">

<p><u> dibawah ini adalah bilangan genap antara 0 sampai 50 : </u></p>

<?php

for ($a = 0; $a <= 50; $a += 2)
{
  print("<b> $a </b>");
  print("<br>");
}

?>

</body>
</html>
```

Pertemuan 13

Penggunaan Fungsi pada PHP

Seperti halnya JavaScript, PHP juga mendukung untuk penggunaan Fungsi (*function*). Fungsi adalah sekumpulan kode yang dapat dipanggil kembali. Ketika dipanggil, kumpulan kode – kode tersebut akan dijalankan. Dengan adanya fungsi, kita tidak perlu menuliskan kode yang sama berulang – ulang dan akan mempermudah pembacaan kode program.

Istilah fungsi menyatakan blok kode yang diberi nama. Sebuah fungsi dapat melakukan suatu peng-*eksekusi*-an sekumpulan kode yang mempunyai kegunaan berbeda. Sebuah fungsi juga dapat menghasilkan suatu nilai berdasarkan proses yang berada di dalamnya. Bentuk pendefinisian sebuah fungsi pada PHP adalahs ebagai berikut :

```
function nama_fungsi (parameter_parameter);
{
  pernyataan_1;
…
  pernyataan_n;
}
```

Contoh fungsi sederhana:

```
<?php

function jumlah($a, $b)
{
$hasil = $a + $b;

return($hasil);
}

?>
```

Pada contoh di atas pernyataan **$hasil = $a + $b;** menjumlahkan kedua parameter yaitu **$a** dan **$b** dan memberikan hasilnya ke dalam variable **$hasil**.

Sedangkan pada pernyataan **return($hasil);** **return** digunakan untuk memberikan nilai balik fungsi. Perlu diketahui bahwa pernyataan return akan membuat fungsi berakhir dan menghasilkan nilai yang sesuai dengan isi variabel **$hasil**.

Contoh fungsi sederhana :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

function jumlah($a, $b)
{
$hasil = $a + $b;

return ($hasil);
}

$c = jumlah (2010, 2);

print($c);

?>

</body>
</html>
```

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$z = 2010;

function tambah_satu($z)
{
$z ++;
}

tambah_satu(&$z);          //tanda & digunakan untuk mereferensikan variabel

echo $z;

?>

</body>
</html>
```

Contoh fungsi sederhana dengan array :

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$angka = array(1,2,3,4,5,6,7,8,9,10);

function genap($pisah)
{
return($pisah % 2 == 0);
}

function ganjil($pisah)
{
return($pisah % 2 == 1);
}

print("<b>Fungsi PHP untuk memisahkan bilangan genap dan ganjil dalam array</b>");
print("<br>");
print("<p><u>Bilangan genap : </u></p>");
print_r(array_filter($angka, "genap"));
print("<br>");
print("<p><u>Bilangan ganjil : </u></p>");
print_r(array_filter($angka, "ganjil"));

?>

</body>
</html>
```

Contoh fungsi dengan Referensi sederhana:

```
<html>
<head>
<title> Belajar PHP </title>
</head>

<body>

<?php

$a = 10;
$b = 20;

function tukar(&$x, &$y)
{
 $z = $x;
 $x = $y;
 $y = $z;
}

print("Kondisi semula adalah : ");
printf("a = %d b = %d <br>" , $a, $b);

tukar($a, $b);

print("Kondisi setelah ditukar : ");
printf("a = %d b = %d <br>" , $a, $b);

?>

</body>
</html>
```