BAB VI
STATEMENT KONTROL

Statement IF …

Sintaks dari statement tersebut dalam PHP adalah

```
if (syarat)
{
      statement1;
      statement2;
      .
      .
}
```

Untuk menyatakan syarat, biasanya digunakan operator perbandingan seperti yang telah dibahas sebelumnya. Apabila syarat bernilai TRUE maka statement-statement yang diapit dengan tanda kurung kurawal akan dijalankan. Bentuk lain dari sintaks IF adalah

```
if (syarat)
{
      statement1;
      statement2;
      .
      .
}
else
{
      statement3;
      statement4;
      .
      .
}
```

Untuk sintaks kedua di atas, statement3, statement4, dst akan dijalankan apabila syarat bernilai FALSE.

Contoh:

```
<?
$my_name = "nada";

if ($my_name == "nada")
{
    echo "Your name is ".$my_name."!"<br>";
}
echo "Welcome to my homepage!";
?>
```

Contoh:

```
<?
$number = 3;

if ($number == 4)
{
      echo "Benar";
}
else
{
      echo "Salah";
}
?>
```

Terdapat pula bentuk sintaks berikutnya dari IF … yaitu dengan ditambahkan `elseif`

```
if (syarat1)
{
      statement11;
      statement12;
      .
      .
}
elseif (syarat2)
{
      statement21;
      statement22;
      .
      .
}
.
.
else
{
      statement1;
      statement2;
      .
      .
}
```

Jika syarat1 bernilai TRUE, maka statement11, statement12 dst akan dijalankan. Sedangkan jika syarat1 FALSE maka selanjutnya akan dicek untuk syarat2. Jika syarat2 TRUE maka statemen21, statement22, dst akan dijalankan, sedangkan jika syarat2 FALSE akan dicek syarat berikutnya (jika masih ada). Statement1, statement2, dst baru akan dijalankan apabila semua syarat sebelumnya bernilai FALSE.

Contoh:

```
<?
$karyawan = "Bob";
if($karyawan == "Tanner")
{
      echo "Hello Tanner!";
}
elseif($karyawan == "Bob")
{
      echo "Hello Bob!";
```

```
}
else
{
      echo "Hello!";
}
```

Statement SWITCH

Sintaks dari statement ini adalah

```
switch (variabel)
{
      case  option1:
            statement11;
            statement12;
            .
            .
            break;
      case  option2:
            statement21;
            statement22;
            .
            .
            break;
      .
      .
      default:
            statementdefault1;
            statementdefault2;
            .
            .
            break;
}
```

Pada sintaks di atas, nilai dari variabel akan dicek pada setiap option yang ada (terletak di bagian case). Jika ada option yang sama dengan nilai variabel, maka statement-statement di bawah option tersebutlah yang akan dijalankan. Bagian default adalah optional (boleh ada, boleh tidak).

Contoh:

```
<?php
$tujuan = "Tokyo";
echo "Biaya Perjalanan Menuju $tujuan adalah ";
switch ($tujuan){
      case "Las Vegas":
            echo " $500";
            break;
      case "Amsterdam":
            echo " $1500";
            break;
      case "Egypt":
            echo " $1750";
            break;
      case "Tokyo":
            echo " $900";
            break;
```

```
      case "Caribbean Islands":
            echo " $700";
            break;
}
?>
```

Contoh:

```
<?php
$tujuan = "New York";
echo "Biaya Perjalanan Menuju $tujuan adalah ";
switch ($tujuan){
      case "Las Vegas":
            echo " $500";
            break;
      case "Amsterdam":
            echo " $1500";
            break;
      case "Egypt":
            echo " $1750";
            break;
      case "Tokyo":
            echo " $900";
            break;
      case "Caribbean Islands":
            echo " $700";
            break;
      default:
            echo " $100";
            break;
}
<?
```

Statement WHILE

Statement ini digunakan untuk mengerjakan suatu statement secara berulang-ulang sampai suatu syarat dipenuhi. Sintaksnya adalah

```
while (syarat)
{
      statement;
      statement;
      .
      .
}
```

Pada sintaks di atas, selama syarat bernilai TRUE maka statement-statement di dalam while akan terus dijalankan secara berulang-ulang. Perulangan baru akan berhenti apabila syarat bernilai FALSE. Sebelum statement yang diulang-ulang dilakukan, terlebih dahulu akan dicek syarat nya apakah bernilai TRUE atau FALSE. Apabila TRUE maka statement akan dijalankan. Sedangkan apabila FALSE, perulangan akan langsung berhenti. Dengan kata lain, statement dalam WHILE bisa jadi tidak akan pernah dilakukan, yaitu apabila syaratnya langsung bernilai FALSE.

Contoh:

```
<?
$harga_sikat = 1500;
$jumlah_sikat = 10;

echo "<table border=\"1\" align=\"center\">";
echo "<tr><td><b>Jumlah Sikat</b></td>";
echo "<td><b>Harga</b></tr></td>";
while ( $jumlah_sikat <= 100 )
{
      echo "<tr><td>";
      echo $jumlah_sikat;
      echo "</td><td>";
      echo "Rp. ".$harga_sikat * $jumlah_sikat;
      echo "</td></tr>";
      $jumlah_sikat = $jumlah_sikat + 10;
}
echo "</table>";
?>
```

Kode di atas akan menampilkan hasil di browser berupa tabel yang berisi jumlah sikat dan harganya, dengan asumsi harga sebuah sikat adalah Rp. 1.500. Jumlah sikat yang ditampilkan adalah kelipatan 10 dengan batas sampai 100 buah.

Statement FOR

Statement FOR mirip dengan WHILE yang memiliki sintaks berikut ini

```
for (inisialisasi counter; syarat; increment/decrement counter)
{
      statement;
      .
      .
}
```

Untuk memperjelas pemahaman tentang FOR, berikut ini adalah contoh kode dengan for untuk menghasilkan tampilan yang sama dengan contoh while sebelumnya (tentang jumlah sikat dan harganya). Coba bandingkan dengan kode contoh while sebelumnya.

Contoh:

```
<?
$harga_sikat = 1500;

echo "<table border=\"1\" align=\"center\">";
echo "<tr><td><b>Jumlah Sikat</b></td>";
echo "<td><b>Harga</b></td></tr>";
for ($jumlah_sikat = 10; $jumlah_sikat <= 100; $jumlah_sikat+=10)
{
      echo "<tr><td>";
      echo $jumlah_sikat;
      echo "</td><td>";
      echo "Rp. ".$harga_sikat * $jumlah_sikat;
      echo "</td></tr>";
}
```

```
echo "</table>";
?>
```

Statement Foreach

Misalkan Anda punya data berupa array assosiatif yang akan diproses secara berulang-ulang, maka PHP menyediakan statement foreach yang mudah digunakan.

Sintaksnya adalah:

```
foreach(variabelarray as kunci => value)
{
      statement;
      .
      .
}
```

Sebagai contoh, misalkan Anda memiliki 5 orang karyawan dengan usianya masing-masing yang ditulis dalam kode PHP sebagai berikut

```
$UsiaKaryawan["Lisa"] = "28";
$UsiaKaryawan["Jack"] = "16";
$UsiaKaryawan["Ryan"] = "35";
$UsiaKaryawan["Rachel"] = "46";
$UsiaKaryawan["Grace"] = "34";
```

Berikut ini adalah contoh kode PHP yang akan menampilkan semua karyawan beserta usianya dengan menggunakan foreach.

```
<?
$UsiaKaryawan["Lisa"] = "28";
$UsiaKaryawan["Jack"] = "16";
$UsiaKaryawan["Ryan"] = "35";
$UsiaKaryawan["Rachel"] = "46";
$UsiaKaryawan["Grace"] = "34";

foreach($UsiaKaryawan as $Nama => $umur)
{
      echo "Nama Karyawan: $Nama, Usia: $umur"." th <br>";
}
?>
```

Statement DO WHILE

Statement ini merupakan bentuk modifikasi dari WHILE. Sintaksnya adalah sebagai berikut

```
do
{
      statement;
      .
      .
}
while (syarat);
```

Coba bandingkan dengan sintaks WHILE sebelumnya. Dilihat dari posisi statement yang diulang, posisi statement yang diulang pada DO WHILE terletak di atas syarat. Dengan demikian, sebelum syarat dicek TRUE atau FALSE nya, statement akan dikerjakan terlebih dahulu. Sedangkan pada WHILE, sebelum statement yang diulang dikerjakan, terlebih dahulu syarat akan dicek.

Prinsip kerja DO WHILE sama dengan WHILE yaitu statement akan terus dikerjakan selama syarat bernilai TRUE dan perulangan akan berhenti apabila FALSE.

Perhatikan contoh berikut ini yang membandingkan DO WHILE dengan WHILE

Contoh:

```
<?
$kue = 0;
while($kue > 1)
{
      echo "Mmmmm...Aku suka kue! Nyam..nyam..nyam..";
}
?>

<?
$kue = 0;
do
{
      echo "Mmmmm... Aku suka kue! Nyam..nyam..nyam..";
} while ($kue > 1);
?>
```

Pada kode WHILE, teks "Mmmm…. " dst tidak akan ditampilkan karena syaratnya langsung bernilai FALSE (perulangan berhenti). Sedangkan pada DO WHILE, teks akan ditampilkan dahulu kemudian perulangan berhenti (syarat bernilai FALSE).