**Kapitel 4: Operatorer och vektorer.**
Du kommer ofta behöva jämföra saker i PHP, för att kunna detta behöver du ha kännedom om operatorerna som finns inbyggda i PHP.
I denna del kommer vi att gå igenom operatorer. Vi kommer även att kika lite på villkorsatser, eftersom de används av operatorerna. Har du ytterst lite koll på hur en villkorssats fungerar kan du läsa vidare, annars kan du kika lite på Del 5 innan du läser denna.

Om du däremot känner dig manad att fortsätta med operatorerna och villkorssatserna, är det bara att läsa vidare!

## Operatorer

I PHP finns följande operatorer (jag kommer inte lista alla)

**Relationsoperatorer (Jämförande operatorer)**

| Operator | Betydelse |
|---|---|
| < | Är mindre än |
| <= | Är mindre än eller lika med |
| === | Är lika med och av samma typ |
| > | Är större än |
| >= | Är större än eller lika med |
| != | Är inte lika som |
| !== | Är inte lika eller tillhör inte samma typ |
| <> | Mer eller mindre än |
| == | Är lika med |

**Logiska operatorer**

| Operator | Betydelse |
|---|---|
| && | Och |
| and | Och |
| \|\| | Eller |
| or | Eller |
| XOR | Exklusivt eller |
| ! | Inte |

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

**Aritmetiska operatorer**

| Operator | Betydelse |
|---|---|
| + | Addition |
| - | Subtraktion |
| * | Multiplikation |
| / | Division |
| % | Modulus |
| -x | Negligering av variabeln x (i det här fallet) |

**Tilldelningsoperatorer**

| Operator | Betydelse |
|---|---|
| = | Tilldela |
| .= | Lägg på ett värde på en redan tilldelad variabel |

Skillnaden mellan and och && bland de logiska operatorerna är att && har högre prioritet, vilket gäller även || och or. Ordningen är som följer: &&, ||, and och sedan or.

Om vi ska lista upp sanningstabeller över de olika logiska operatorerna.

0 = Falskt  1 = Sant

**Sanningstabell 1**

| P | Q | P && Q |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

&& Returnerar TRUE om både P och Q är sanna.

**Sanningstabell 2**

| P | Q | P \|\| Q |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

|| Returnerar TRUE om något av P och Q är sanna.

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

**Sanningstabell 3**

| P | P ! Q |
|---|---|
| 1 | 0 |
| 0 | 1 |

! Returnerar TRUE om P är falskt.

**Sanningstabell 4**

| P | Q | P XOR Q |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

XOR Returnerar TRUE om antingen P eller Q är sanna, men inte om både är sanna.

Principen för hur en relationsoperator kollar kan se ut såhär:

```
# KODSTYCKE 11 - OPERATORPRINCIP.PHP
```

```
<?PHP
if (villkor)

{
     Om det returnerar true hamnar man här.
}
?>
```

Om vi tar och ställer ett riktigt villkor då.

```
# KODSTYCKE 12 - OPERATORTEST.PHP
```

```
<?PHP
if (10 > 9)

{
     echo "10 är större än 9";
}

else

{
     echo "9 är större än 10";
}
?>
```

Vi använde oss av en relationsoperator. Vi ska nu använda både relationsoperatorer och logiska operatorer i samma frågesats.

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

```
# KODSTYCKE 13 - OPERATORTEST2.PHP
<?PHP
if ( (10 > 9) && (11 > 10) )
{
     echo "10 är större än 9 och 11 är större än 10";
}
else
{
     echo "9 är större än 10 och 10 är mindre än 11";
}
?>
```

Principen för att använda logiska operatorer blir alltså:

```
# KODSTYCKE 14 - OPERATORPRINCIP2.PHP
<?PHP
if ( (villkor) && (villkor) )
{
     Båda villkoren returnerade true
}
else
{
     Något av villkoren returnerade false
}
?>
```

Anledningen till att vi använder dubbla paranteser i kodstycke 14, är att det är lättare att hålla isär alla villkor när man separerar dem med flera paranteser.

## Vektorer

En vektor är väldigt lik en variabel, den har egentligen samma uppgift, att lagra information.
Låt oss säga att vi vill lagra mycket information, det blir på tok för omständigt att spara dem i olika variabler, visst går det, men det är dumt. Det är där vektorer kommer in, en vektor är en variabel som man kan lagra mycket information i.
Till skillnad från vanliga variabler kan man stoppa in värde efter värde i en och samma vektor.

Jag ska gå igenom hur man skapar en vektor och tilldelar värden i den

```
# KODSTYCKE 15 - ARRSET.PHP
<?PHP
$arrArray = array("PHP","ASP","JSP");
?>
```

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

Detta skulle då skapa en vektor med nycklarna 0, 1 och 2. PHP börjar alltid tilldelningen av nycklar med 0, om man inte anger något annat d v s.

Ett annat sätt att tilldela en vektor värden på, är att använda sig av hakparanteser ([]), här kan vi även styra nyckeltilldelningen genom att ange ett värde inom hakparantesen

# KODSTYCKE 16 - ARRSET2.PHP

```
<?PHP
$arrArray[1] = "PHP";
$arrArray[] = "ASP";
$arrArray[] = "JSP";
?>
```

Med detta har vi då skapat nycklarna 1, 2 och 3 med likadana värden som i Kodstycke 15.

Visserligen kan vi styra nyckeltilldelningen genom array() funktionen som vi använde i Kodstycke 15 också, genom följande kod genereras samma vektor som i Kodstycke 16

# KODSTYCKE 17 - ARRSET3.PHP

```
<?PHP
$arrArray = array(1 => "PHP", "ASP", "JSP");
?>
```

Men, vi vill inte ha siffror som nycklar, vi vill kanske ha en sträng som nyckel, visst. Det går det också.

# KODSTYCKE 18 - ARRSET4.PHP

```
<?PHP
$arrArray = array(
     "PHP" => "Hypertext Preprocessor",
     "ASP" => "Active Server Pages",
     "JSP" => "Java Server Pages"
);
?>
```

Då har vi skapat nycklarna PHP, ASP och JSP. Vi kan nu komma åt dessa värden genom att skriva följande:

# KODSTYCKE 18 - ARRPRINT.PHP

```
<?PHP
$arrArray = array(
     "PHP" => "Hypertext Preprocessor",
     "ASP" => "Active Server Pages",
     "JSP" => "Java Server Pages"
);
echo $arrArray['PHP'];
?>
```

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

Då skulle vi få ”Hypertext Preprocessor” utskrivet på skärmen. På samma sätt som i Kodstycke 18 kan vi komma åt de övriga nycklarna JSP och ASP.

För att få fram alla nycklar och värden i en vektor kan man använda sig av en funktion som heter print_r() på följande sätt

```
# KODSTYCKE 17 - ARRPRINT.PHP
<?PHP
$arrArray = array(1 => "PHP", "ASP", "JSP");
print_r($arrArray);
?>
```

Då skulle vi få följande utskrift

```
# KODSTYCKE 18 - UTSKRIFT
Array
(
    [1] => PHP
    [2] => ASP
    [3] => JSP
)
```

Detta är ett mycket effektivt sätt för att få fram alla värden i en vektor.

**Göra en slinga**

Även om vi inte kommit in på slingor ännu i kursen så tänkte jag gå igenom här hur man gör för att skriva ut vektorn med lite mer ordning än när vi använde print_r(), vi ska använda oss av foreach() slingan, den går igenom en vektor element efter element tills den gått igenom hela vektorn.
Vi tar följande exempel.

```
# KODSTYCKE 18 - ARRLOOP.PHP
<?PHP
$arrArray = array("PHP", "ASP", "JSP");
echo "Här är några av de olika server-baserade språken<br /> ";
foreach ($arrArray as $key)
{
    echo $key . "<br /> ";
}
?>
```

Detta skulle ge utskriften

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

# KODSTYCKE 19 - UTSKRIFT2

```
Här är några av de olika server-baserade språken
PHP
ASP
JSP
```

foreach() slingan är uppbyggd enligt följande princip

# KODSTYCKE 20 - FOREACHPRINCIP.PHP

```
foreach ($vektor as $värde)
{
     utskrift
}
```

Man lagrar alltså varje element i $värde och skriver sedan ut det för att gå vidare till nästa element.

Men låt oss nu gå tillbaka till vektorn vi använde i Kodstycke 18 och bygga vidare på det exemplet med foreach() slingan.

# KODSTYCKE 21 - ARRLOOP2.PHP

```
<?PHP
$arrArray = array(
     "PHP" => "Hypertext Preprocessor",
     "ASP" => "Active Server Pages",
     "JSP" => "Java Server Pages"
);
foreach ($arrArray as $key => $value)
{
     echo $key . " betyder " . $value ."<br />";
}
?>
```

Eftersom vi hade nu en nyckel och ett värde, kunde vi använda det i foreach() slingan.

Vi bygger på principen för foreach() lite.

# KODSTYCKE 22 - FOREACHPRINCIP2.PHP

```
foreach ($vektor as $nyckel => $värde)
{
     utskrift
}
```

Detta betyder dock inte att om man bara vill ha värdet och inte nyckeln att man måste skriva ut nyckeln först, det är bara att anropa $värde istället för $nyckel.

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

### Flerdimensionella vektorer

Man kan även göra en vektor flerdimensionell genom att lägga in ytterligare ett vektorselement i en existerande vektor.

Den kan se ut såsom följer

```
# KODSTYCKE 22 - ARRMULTI.PHP
```

```
<?PHP
$arrArray = array(
     "Grafik" => array("Photoshop", "3DSMax", "GIMP"),
     "Editors" => array("Vim", "Maguma studio", "Emacs"),
     "Webbläsare" => array("Firefox", "Internet explorer",
"Mozilla", "Opera")
);
foreach ($arrArray as $key => $value)
{
     echo "<b>" . $key . "</b><br />";
     foreach ($value as $value2)
     {
          echo $value2 . "<br />";
     }
     echo "<br />";
}
?>
```

Som ni ser så har vi byggt in ytterligare ett array() element inuti $arrArray vilket gör att vi får lov att köra ytterligare en foreach() slinga inuti en redan påbörjad foreach() slinga. Vi använder även $value värdet som vektor när vi påbörjar nästa foreach() slinga.

```
# KODSTYCKE 23- UTSKRIFT3
```

```
Grafik
Photoshop
3DSMax
GIMP

Editors
Vim
Maguma studio
Emacs

Webbläsare
Firefox
Internet explorer
Mozilla
Opera
```

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.

## Uppgifter

Vi kommer även i denna del att ge uppgifter inför nästa del.

### Operatorer

1. Du ska skapa en sida som räknar ut summan av 100+24 och sedan delar det med 40.
2. Samt att du ska skapa en frågesats som kollar om 9 är större än 10.

### Vektorer

1. Skapa en vektor som innehåller alla Sveriges större städer och skriver ut dem på skärmen i en fin ordning.
2. Skapa en vektor som innehåller en lista på behändiga verktyg att ha på sin arbetsstation, lös det antingen med flerdimensionella vektorer eller utan.

## Lösning från del 3

```
# SIDA1.PHP - Uppbyggd med variabler
```

```
<?PHP
$fornamn = "Andreas";
$efternamn = "Jönsson";
$mittnamn = $fornamn . " " . $efternamn;
echo $mittnamn;
?>
```

Notera att vi använder ”.” konstruktionen i exemplet.

```
# SIDA2.PHP - Uppbyggd med konstanter
```

```
<?PHP
define("fornamn", "Andreas");
define("efternamn"," Jönsson");
echo fornamn . " " . efternamn;
?>
```

Kom ihåg att du måste skriva konstanten exakt som du definerade den, annars kommar den inte att skrivas ut.

Det finns givetvis övriga sätt att göra detta på, men detta är sättet jag kommer att lära ut i denna kurs.

©2002 - 2005 PHPSidan (http://www.phpsidan.nu)
Allt material är framställt av PHPSidan. All kopiering strängeligen förbjuden.