PHP – En överlevnadsguide del 1

Vad är PHP?

PHP (**P**HP **H**ypertext **P**reprocessor) är ett skriptspråk som körs på webbservrar (**Server-Side**) för att generera **dynamiskt** innehåll samt kommunicera med databaser t.ex. my**SQL**.

Grunderna i PHP

Filerna bör ha filändelsen **.php** för att webbservern ska förstå att det är php-kod som ska tolkas.

Det går alldeles utmärkt att blanda **HTML** och **PHP**. Grundregeln är att innehåll som är **statiskt** (inte ändras beroende på input eller liknande) ska vara HTML samt försöka minimera mängden HTML-kod som genereras av PHP.

PHP-kod infogas med `<?php` taggen och avslutas med `?>`

Exempel:

```
<?php
   //Detta är en kommentar
?>
```

Ett kommando eller instruktion avslutas med tecknet `;` en kommentar inleds med `//`

Skriva ut text

För att skriva ut text används kommandot `echo` eller `print` (fungerar på samma sätt).

Exempel:

```
echo "Hej";
print " Kalle!";
```

Resultat: Hej Kalle!

Observera att det inte blir någon ny rad mellan "Hej" och "Kalle".

Använda en inbyggd funktion

Det finns tusentals inbyggda funktioner i PHP och PHPs vanliga s.k. extensions (tillägg). Därför är det omöjligt att kunna alla dessa utantill. Det viktiga är att kunna söka, hitta och använda lämpliga funktioner beroende på tillämpning. Det mesta hittar man i manualen för PHP. En vanligt förekommande funktion är `date()` som ger oss aktuellt datum och/eller tid i olika format.

Exempel:

```
echo date('Y-m-d');
//Visar dagens datum
```

Resultat: 2014-10-02

Slå samman textsträngar

Väldigt ofta vill man slå samman textsträngar till längre texter. Detta görs med punkttecknet `.` (concatenation operator). Fungerar även bra med variabler.

Exempel:

```
echo "Hej " . "Då!";
```

Resultat: Hej Då!

Variabler

En **variabel** kan ses som en behållare med ett namn som innehåller något värde. En variabel inleds med `$` tecknet. Variabelnamn är **case-sensitive**. Det finns reserverade specialnamn som ej kan användas för egna variabelnamn (t.ex. $this). När man **deklarerar** en variabel så har den inget värde förrän den **tilldelats** ett värde, detta görs med `=` tecknet (tilldelningsoperatorn). En variabel kan inte användas förrän den tilldelats ett värde.

Exempel:

```
$minVariabel = 5;
echo $minVariabel;
```

Resultat: 5

PHP-språket är ett s.k. löst typat språk vilket innebär att man inte behöver tala om vilken typ (heltal eller text mm.) variabeln ska vara. En variabel kan alltså vara heltal, decimaltal, text, ett fält (array) eller ett objekt mm. och kommer att behandlas olika beroende på vad PHP tror att det är eller ska vara. Det finns sätt att tvinga PHP att behandla en variabel på ett visst sätt vid tvivel (mer om det senare).

Skillnaden mellan " och '

Citattecken används normalt för att ange något i text-format. Ibland så spelar det ingen roll vilken typ av citattecken som används men i vissa fall är det stor skillnad. En grundregel är att inte blanda ihop olika typer av citattecken. Ska man ange ett uttryck inom citattecken som i sin tur innehåller citattecken så ska de

PHP – En överlevnadsguide del 1

yttre citattecknen skilja sig från den "inre" typen av citattecken. T.ex. "Yttre uttryck 'inre uttryck' fortsättning yttre".

Exempel:

```
$namn = "Kalle";
echo "Hej $namn <br>";
echo 'Hej $namn';
```

Resultat

```
Hej Kalle
Hej $namn
```

Vid utskrift så kommer print och echo kommandon att tolka s.k. specialtecken (**escape characters**) ifall "-tecknen används. Dessa inleds med tecknet \. Även variabelnamn kommer att tolkas och ersättas med variabelns värde. Använder vi däremot '-tecken så kommer det vi ange att tolkas slaviskt/bokstavligt (se exemplet ovan).

Räkneoperatorer

När vi programmerar PHP så kan vi använda de vanliga räkneoperatorerna för addition +, subtraktion -, multiplikation *, och division /. Detta fungerar precis som i en miniräknare och vi kan även använda parenteser.

Exempel:

```
$tal = (5 + 3) / 2;
$tal2 = $tal – 2 * 2;
```

I exemplet ovan får $tal värdet 4 och $tal2 får värdet 0 (vanliga räkneregler gäller).

Jämförelseoperatorer och logik

För att kunna göra jämförelser och ställa upp logiska uttryck så används följande operatorer

Operator	Betydelse
==	Lika med
<	Mindre än
>	Större än
<=	Mindre än eller lika med
>=	Större än eller lika med
!=	Skiljt från
&&	OCH (AND)
||	ELLER (OR)
and	Samma som $$
or	Samma som ||

Flera jämförelser och logiska uttryck kan kombineras med && eller ||. Ett logiskt uttryck eller en jämförelse kan antigen resultera i sant (**true**) eller falskt (**false**). I PHP hanteras dessa booleska värden oftast som 1 och 0 eller null när de omvandlas.

Exempel:

```
$bool1 = true;
$bool2 = 5>7;
$bool3 = $bool1 && (5<6);
echo $bool3;
```

Resultat

```
1
```

I exemplet ovan ges variabeln $bool1 värdet true som är ett booleskt värde. $bool2 får värdet false och $bool3 får värdet true vilket skrivs ut som 1. Ett false-värde skrivs ut som **null** (inget alls). PHPs hantering av typkonvertering och booleska värden i vissa situationer har kritiserats mycket av erfarna programmerare.

IF-sats

För att styra vad som ska hända så kan vi använda en villkorssats. Den vanligaste typen är IF-satsen. Om (IF) ett villkor eller logiskt uttryck stämmer så körs ett block med kod annars körs det inte. Tänk på att det går att kombinera flera villkor och på så sätt skapa avancerade uttryck.

Exempel:

```
$age = 25;
if ($age >= 18) {
   echo "Du är myndig";
}
```

Resultat

```
Du är myndig
```

Det går givetvis att lägga vad man vill i kodblocket, t.ex. fler IF-satser.

IF-ELSE

Man kan komplettera IF-satsen med ett ELSE-block som enbart körs då villkoret för IF-satsen inte stämmer (= false).

Exempel:

```
$age = 15;
if ($age >= 18) {
   echo "Du är myndig";
}
else {
   echo "Du är inte myndig";
}
```

Resultat

```
Du är inte myndig
```

PHP – En överlevnadsguide del 1

Det går även att komplettera ELSE med ELSE-IF som är ännu ett villkor som kontrolleras ifall första IF-satsens villkor är falskt.

Exempel:

```
$age = 16;
if ($age >= 18) {
   echo "Du är myndig";
}
else if ($age >15){
   echo "Du är snart myndig";
}
```

Resultat

Du är snart myndig

SWITCH-sats

Vill man göra olika saker beroende på t.ex. ett värde på en variabel så kan man lösa detta med flera IF-satser. Ett snyggare och bättre sätt är dock oftast att använda en SWITCH-sats. Förutom talvärden så går det bra att

Exempel:

```
$i = "a";
switch ($i) {
   case "a":
   case "b":
      echo "i är a eller b";
      break;
   case "z":
      echo "i är z";
      break;
   default:
      echo "ingen matchning";
}
```

använda textsträngar (som i exemplet).

Fält (array)

En **array** eller **fält** på svenska är ett antal element samlat i en datastruktur. Varje element har ett värde. Elementens värden kan vara av olika typ vilket skiljer sig från hårt typade språk som Java och C#. Varje element får ett nummer, ett s.k. **index** som vi kan använda för åtkomst till ett element i fältet. **Indexeringen börjar alltid på 0** och är ett heltal. Ett index anges antigen med `[]` eller `{}`. I PHP kan fält även indexeras med textsträngar (mer om detta senare). En array kan skapas med `array()`.

Exempel:

```
$test = array("m", "ja", "no");
echo $test[0];
echo " " . $test{2};
```

Resultat

m no

Ta bort fält och element i fält

För att ta bort ett helt fält eller ett element i ett fält används den inbyggda funktionen `unset()`. Ta bort variabler fungerar på samma sätt.

Exempel:

```
$test = array("a", "b", "c");
unset($test[1]);
echo $test[1];
```

Resultat

Error! Odefinierat index

I exemplet ovan tas index 1 bort ur fältet. Index 0 och 2 finns fortfarande kvar.