

Web Programmering med Arrays i PHP

Jonas Holbech og Martin Elsmann
IT Universitetet i København*

24. september 2007

Resumé

I denne note gives en introduktion til brugen af arrays i PHP. Notens beskriver hvordan arrays oprettes og benyttes. Herudover beskrives en række simple funktioner til manipulation af arrays i PHP.

1 Introduktion

Vi har tidligere set hvorledes variabler i PHP kan benyttes til at indeholde værdier, såsom heltal og strenge. I denne note introducerer vi en ny slags værdi kaldet et *array*. Et *array* er en værdi som kan indeholde andre værdier som elementer; disse elementer er værdier, der som hovedregel er af samme type, men ikke nødvendigvis. I PHP er hvert element i et array *indekseret* med enten et heltal eller med en streng. Til at starte med vil vi se på arrays som er indekseret med heltal, kaldet *tal-indekserede arrays*, hvorefter vi vil se på arrays indekseret med strenge, kaldet *streng-indekserede arrays*.

2 Tal-Indekserede Arrays

Hvis man i PHP skal skrive kode til at huske en række brugernavne kan man gemme hvert navn i en variabel og udskrive indholdet af variablerne en af gangen, som følger:

```
function printName($name) {  
    echo "Navnet er $name";  
}  
  
$name1='Ole';  
$name2='Lis';  
$name3='Martin';
```

*Adresse: Rued Langgaards Vej 7, 2300 København S, Danmark. Email: holbech@itu.dk og mael@itu.dk.

```
printName($name1).'<br />';  
printName($name2).'<br />';  
printName($name3).'<br />';
```

Kørsel af ovenstående kode resulterer i følgende HTML-output:

```
Navnet er Ole<br />  
Navnet er Lis<br />  
Navnet er Martin<br />
```

Denne løsning er dog problematisk hvis antallet af navne der skal udskrives ikke kendes på forhånd og samtidig kræver løsningen at der ved ændringer i listen af navne ændres både i udskrivningskoden samt i koden for at oprette navnene. Med arrays og vores viden om `while`- og `for`-løkker kan ovenstående kode skrives smartere, herunder kortere, og koden kan gøres mere dynamisk.

Traditionelt indekseres arrays med fortløbne tal, startende med 0. Visuelt kan sådanne tal-indekserede arrays præsenteres som følger:

Indeks	Værdi
0	Ole
1	Lis
2	Martin
3	Jan

2.1 Arrays i PHP

I PHP kan arrays grundlæggende oprettes ved brug af `array()` syntaksen eller ved brug af `[]` operatoren.

Følgende PHP kode opretter et array med fem streng elementer og tilegner værdien til variabelen `$students`:

```
// opret et array med fem elementer  
$students = array("Mads", "Lars", "Jørgen", "Bent", "Ulla");
```

For, for eksempel, at tilgå det fjerde element i arrayet, "Bent", skal indekset 3 benyttes, da det første element tilgås ved at benytte indeks 0:

```
echo $students[3]; //udskriver Bent
```

Generelt, for et array med N elementer, tilgås det sidste element ved at benytte indeks $N - 1$.

Det er muligt at oprette og tilføje elementer til et eksisterende array ved hjælp af `[]` operatoren. Nedenfor ses en alternativ måde at oprette arrayet i variabelen `$students` på:

```
// et tomt array oprettes og tildeles variabelen $students
$students = array();

// elementerne tilføjes et ad gangen
$students[] = "Mads";
$students[] = "Lars";
$students[] = "Jørgen";
$students[] = "Bent";
$students[] = "Ulla";
```

Ligesom i den første metode til konstruktion af arrayet angives arrayets indekser ikke under oprettelsen af arrayet. Som tidligere kan værdien “Bent” tilgås ved at benytte indekset 3.

Det er tilladt at angive indekset eksplicit ved opdatering af et array; herved kan rækkefølgen af indsættelserne gøres uafhængig af hvor i arrayet de enkelte elementer placeres. Således kan arrayet i variabelen `$students` ovenfor også oprettes som følger:

```
$students = array();
$students[0] = "Mads"
$students[1] = "Lars";
$students[3] = "Bent";
$students[2] = "Jørgen";
$students[4] = "Ulla";
```

Ved indeksering af arrays er der ingen tvang om at bruge fortløbne numre. Nye elementer kan lovligt tilføjes som følger:

```
$students[666] = "Jonas"
$students[999] = "Martin";
```

Det er dog et krav at der kun benyttes positive tal.

Operatoren `[]` benyttes ligeledes til at overskrive eksisterende elementer i et array. Indholdet af et array kan derfor ændre sig under afvikling (kørsel) af et program (PHP-kode). Følgende kode viser et eksempel på dette:

```
// et array oprettes med tre elementer
$positions = array("Programmør", "Grafisk designer", "Projektleder");

// projektlederstillingen er trukket tilbage, men der er kommet
// en ny stilling, så det tredje element overskrives.
$positions[2] = "Adm. Direktør";
```

Arrayet `$positions` indeholder nu:

Indeks	Værdi
0	Programmør
1	Grafisk designer
2	Adm. Direktør

Antag nu at følgende kode køres:

```
// tilføj en ny stilling
$positions[] = "Freelance";
```

Det nye element bliver indekseret med “det næste frie nummer”, hvorefter arrayet ser således ud:

Indeks	Værdi
0	Programmør
1	Grafisk designer
2	Adm. Direktør
3	Freelance

3 Streng-Indekserede Arrays

I PHP er det tilladt at benytte sig af strenge til indeksering af arrays. Arrays hvor strenge er benyttet til array-indeksering kaldes streng-indekserede arrays, som ofte også kaldes *associative arrays*. Hvor tal-indekserede arrays er lette at manipulere, for eksempel ved at nye elementer kan tilføjes uden at vi skal bekymre os om hvad næste indeks er, er kode med brug af tal-indekserede arrays ikke altid let at læse.

Nedenfor ses et eksempel på oprettelse af et streng-indekseret array:

```
// opret et associativt array
$customerOrder = array("itemcode" => "sw345q", "itemname" =>
    "Sække stol", "color" => "Grøn");
```

Ovenstående kan med fordel (for at gøre koden lettere at læse) formateres som følger:

```
$customerOrder = array(
    "itemcode" => "sw345q",
    "itemname" => "Sække stol",
    "color"     => "Grøn"
);
```

Bemærk at når elementerne oprettes benytter vi tegnene => til at angive hvad der er indeks og hvad der er elementets værdi.

Ligesom tal-indekserede arrays refererer vi elementer i arrayet ved at benytte indekset. Vil vi have fat i tredje element (ordrens farve) skriver vi eksempelvis:

```
echo $customerOrder["color"];
```

På samme måde som med tal-indekserede arrays, kan elementer tilføjes og overskrives:

```
// skift af farven på sækkestolen
$customerOrder["color"] = "Blå";

// detaljer til ordren tilføjes
$customerOrder["price"] = "1200";
$customerOrder["size"] = "XL";
```

For både tal-indekserede og streng-indekserede arrays gælder det at indholdet af en variabel kan benyttes til at angive indekset og/eller værdien af et element:

```
// opret nogle variabler
$pen = "Kuglepen";
$coins = "5,25 i assorterede mønter";
$paper = "papir";

// opret et tal-indekseret array
$itemsInMyBag = array("computer", "usb pind", "bog", $pen);

// udskriv de to sidste elementer
echo "<h1>Det tal-indekserede array</h1>";
echo "<p>${itemsInMyBag[2]}</p>";
echo "<p>${itemsInMyBag[3]}</p>";

// opret et streng-indekseret array -- overskriv det gamle
$itemsInMyBag = array(
    "coins" => $coins,
    "computer" => "Vega 31M+",
    $paper => "linieret"
);

// tilføj et ekstra element
$itemsInMyBag[$pen] = $pen;

// udskriv alle elementerne, en ad gangen -- dette
// gøres lidt smartere om lidt
echo "<h1>Det streng-indekserede array</h1>";
echo "<p>Penge: ".$itemsInMyBag['coins']. "</p>";
echo "<p>Computer: ".$itemsInMyBag['computer']. "</p>";
echo "<p>${paper}: ".$itemsInMyBag[$paper]. "</p>";
echo "<p>${pen}: ".$itemsInMyBag[$pen]. "</p>";
```

Kørsel af ovenstående kode resulterer i følgende output:

```

<h1>Det tal-indekserede array</h1>
<p>bog</p>
<p>Kuglepen</p>
<h1>Det streng-indekserede array</h1>
<p>Penge: 5,25 i assorterede mønter</p>
<p>Computer: Vega 31M+</p>
<p>$paper: linieret</p>
<p>$pen: Kuglepen</p>

```

Læg mærke til at koden benytter sig af variabler til både at definere indeks-værdier (nøgler) og element-værdier. I eksemplet er `$paper` sat til at indeholde streng-værdien "papir". Et par linier nedenunder variabel-erklæringen benyttes variabelen til at tilføje et element til det associative array:

```
$itemsInMyBag[$pen] = $pen;
```

Ligeledes gør udskrivningskoden brug af indholdet af variabelen `$paper`:

```
echo "<p>$paper: ".$itemsInMyBag[$paper]."</p>";
```

Idet variabelen `$paper`, under udførslen af udskrivningskoden, indeholder strengen "papir", kan udskrivningskoden også forstås således:

```
echo "<p>$paper: ".$itemsInMyBag["papir"]."</p>";
```

3.1 Streng-indekserede eller tal-indekserede arrays

Streng-indekserede arrays er ofte nyttige til lagring af attributter knyttet til en logisk entitet hvor tal-indekserede arrays ofter er anvendelige når noget kode skal holde styr på en sekvens (eller en mængde) af entiteter. Ofte ses kode som udelukkende benytter sig af tal-indekserede arrays selvom tilsvarende kode der gør brug passende streng-indekserede arrays vil være lettere at læse og vedligeholde. Hvis vi således ønsker at samle information om en person i et array `$person`, vil det værre lettere at læse kode der henviser til `$person['email']` end kode der henviser til `$person[0]` med den underliggende antagelse at email-information ligger i element 0 i arrayet.

Omvendt giver for-løkker og while-løkker mest mening i sammenhæng med tal-indekserede arrays.

4 Manipulation af Arrays med Funktioner

Der findes en lang række funktioner i PHP til manipulation af arrays. Vi vil i det følgende berøre nogle af de vigtigste af disse funktioner. Det er vigtigt at gøre sig klart at alle disse funktioner kan skrives fra bunden ved blot at benytte sig af de primitive metoder til at tilgå arrays.

4.1 Funktion til udskrivning af arrays

Hvis man prøver at udskrive et array ved hjælp af `echo` eller `print()` får man ikke nødvendigvis det resultat man forventer. Således udskriver følgende kode blot strengen `Array`:

```
$items = array("stol", "bord", "lampe");  
echo $items;
```

Det er således typen af værdien der udskrives og ikke værdien indeholdende de enkelte elementer. I stedet kan man med fordel benytte funktionen `print_r()`, som tager et array som argument og udskriver alle nøgler (indeks-værdier) og deres element-værdier i arrayet. Således vil koden

```
print_r($items);
```

udskrive følgende:

```
Array ( [0] => stol [1] => bord [2] => lampe )
```

Funktionen `print_r()` kan derfor være god at kende til for at udskrive og se hvilke nøgler og element-værdier et array indeholder.

For eksempel kan `print_r()` funktionen med fordel bruges til at “debugge” POST og GET data, da sådan data placeres i PHP arrays af serveren når denne modtager et request fra en browser. Antag således at følgende kode skrives i starten af en PHP-fil:

```
<? print_r($_REQUEST); ?>
```

Ved kørsel af PHP-scriptet vil indholdet af `$_REQUEST` arrayet blive udskrevet, hvilket indeholder modtaget POST/GET data samt cookie data. For eksempel kunne koden resultere i følgende udskrift:

```
Array ( [orderId] => stol [orderId] => example@example.com )
```

4.2 Funktion til at finde antal elementer i et array

Funktionen `sizeof()` tager et array som argument og returnerer antallet af elementer i arrayet.

Betragt følgende eksempel:

```
$myarray = array("a", "b", "c", "d");  
$numberOfElements = sizeof($myarray);  
echo "Antallet af elementer er $numberOfElements";
```

Ovenstående kode udskriver følgende:

```
Antallet af elementer er 4
```

4.3 Funktion til at teste om en værdi er et array

Funktionen `is_array()` tager en vilkårlig værdi som argument og returnerer `true` (af type `boolean`) hvis værdien er et array og `false` ellers.

4.4 Funktion til at teste om en værdi er i et array

Funktionen `in_array($needle, $haystack)` tager to argumenter, en værdi `$needle` samt et array `$haystack` og returnerer `true` hvis værdien `$needle` findes som element i arrayet `$haystack`. Ellers returnerer funktionen `false`.

Eksempel:

```
$myarray = array("a", "b", "kim", "d");
if ( in_array('kim', $myarray) )
{
    echo "Kim er en del af arrayet \"$myarray\";
}
```

Da strengen "kim" findes i arrayet udskriver koden følgende tekst:

```
Kim er en del af arrayet $myarray
```

5 Arrays og løkker

Vi vil nu se på hvorledes man kan skrive kode til at gennemløbe et array for at behandle elementerne et ad gangen.

Her følger et eksempel på hvorledes kode kan gennemløbe et tal-indeksret array og udskrive HTML-kode for hvert element i arrayet:

```
$persons = array('Jesper', 'Sabina', 'Torill');
$number_of_persons = sizeof($persons); // find antal elementer
$i = 0; // initialiser tæller
while ( $i < $number_of_persons )
{
    echo "<p>$persons[$i]</p>\n"; // udskriv element $i
    $i++; // optæl $i
}
```

Ovenstående kode resulterer i følgende HTML-kode:

```
<p>Torill</p>
<p>Sabina</p>
<p>Jesper</p>
```

Her følger en løkke der resulterer i at elementerne udskrives i omvendt rækkefølge:

```

$i = $number_of_persons - 1;    // start med sidste element
while ( $i >= 0 )              // fortsæt hvis $i > 0
{
    echo "<p>$persons[$i]</p>\n"; // udskriv element $i
    $i--;                       // nedtæl $i
}

```

Nedenfor følger et alternativ til koden som i stedet for en `while`-løkke benytter sig af en `for`-løkke:

```

for( $i = sizeof($number_of_persons) - 1 ; $i >= 0 ; $i++ )
{
    echo "<p>$persons[$i]</p>";
}

```

Endelig kan man gøre brug af PHP-konstruktionen `foreach(){...}` til at gennemløbe et tal-indeksert array. Hvis vi antager at variabelen `$array` indeholder et tal-indeksert array vil konstruktionen kunne benyttes til at behandle hvert element i arrayet:

```

foreach ( $array as $val ){
    // krop
    ...
}

```

Konstruktionen løber gennem elementerne i arrayet, ligesom vi gjorde det tidligere med `while`- og `for`-løkker. For hvert element køres “kroppen” af konstruktionen og det enkelte element stilles til rådighed i kroppen i variabelen `$val`. Her følger et eksempel på brug af konstruktionen:

```

$towers = array( "Sears Tower", "Tower of Hanoi", "Runde Tårn" );
foreach ( $towers as $val )
{
    echo "<p>$val</p>";
}

```

Koden resulterer i følgende HTML-kode når den køres:

```

<p>Sears Tower</p>
<p>Tower of Hanoi</p>
<p>Runde Tårn</p>

```

5.1 Gennemløb af streng-indekserede arrays

For at gennemløbe et streng-indeksert array kan vi igen gøre brug af den indbyggede PHP-konstruktion `foreach(){...}`. Hvis vi antager at variabelen `$array` indeholder et streng-indeksert array vil konstruktionen kunne benyttes til at behandle hvert element i arrayet:

```
foreach ( $array as $key => $val ){
    // krop
    ...
}
```

Som for den simple anvendelse løber konstruktionen igen gennem elementerne i arrayet. For hvert element køres “kroppen” af konstruktionen og det enkelte elements nøgle (indeks-værdi) og element-værdi stilles til rådighed i kroppen i variableerne `$key` og `$val`. Her følger et eksempel på brug af konstruktionen:

```
$countries = array("DK"=>"Danmark", "SE"=>"Sverige",
                  "NO"=>"Norge");
foreach( $countries as $key => $val )
{
    echo "<p>Landekoden for $val er $key</p>";
}
```

Koden resulterer i følgende HTML-kode når den køres:

```
<p>Landekoden for Danmark er DK</p>
<p>Landekoden for Sverige er SE</p>
<p>Landekoden for Norge er NO</p>
```

6 Multidimensionale Arrays

Et array kan indeholde andre arrays hvilket for eksempel kan være praktisk hvis man vil gemme mere end en oplysning om en person og samtidig skal kunne håndtere en mængde af personer. Følgende tabel viser information om tre personer:

Navn	CPR	Gade
Ole	060878-1623	Nørrebrogade
Per	050549-1723	Østerbrogade
Lis	010203-1818	Vesterbrogade

Tabellen viser en liste af personer hver med information om navn, cpr-nummer og gade. Tabellen har således to dimensioner, en dimension der går i retning af antallet af personer i tabellen og en dimension der går i retning af antallet af attributter tilknyttet hver person.

I PHP kan man implementere multidimensionale arrays som arrays indeholdende arrays. I kodeeksemplet nedenfor opretter vi således et multidimensionalt array i PHP og udskriver arrayet som en HTML-tabel:

```
<?
$persons = array(
    array("Ole", "060878-1623", "Nørrebrogade"),
    array("Per", "050549-1723", "Østerbrogade"),
```

```

        array("Lis", "010203-1818", "Versterbrogade")
    );
?>
<table cellpadding='5' border='1'>
<tr><th>Navn</th><th>CPR</th><th>Gade</th></tr>
<?
// først løber vi igennem det ydre array
for( $row = 0 ; $row < sizeof($persons) ; $row++ )
{
    // udskriv hvert af elementerne i det indre array
    echo "<tr>
        <td>" . $persons[$row][0] . "</td>
        <td>" . $persons[$row][1] . "</td>
        <td>" . $persons[$row][2] . "</td>
        </tr>";
}
?>
</table>

```

En kørsel af ovenstående kode resulterer i følgende HTML-kode:

```

<table cellpadding='5' border='1'>
<tr><th>Navn</th><th>CPR</th><th>Gade</th></tr>
<tr><td>Ole</td><td>060878-1623</td><td>Nørrebrogade</td></tr>
<tr><td>Ole</td><td>060878-1623</td><td>Nørrebrogade</td></tr>
<tr><td>Ole</td><td>060878-1623</td><td>Nørrebrogade</td></tr>
</table>

```

Når vi vil tilgå elementer i et multidimensionalt array benytter vi blot operatoren `[]` gentagne gange. For at tilgå Pers gade-navn i `$persons` arrayet ovenfor kan vi skrive

```
$persons[1][2];
```

Med koden `$persons[1]`, fortæller vi PHP at vi er interesserede i det andet element i arrayet `$persons`. Med koden `$persons[1][2]` fortæller vi PHP at vi yderligere er interesserede i det tredje element i arrayet `$persons[1]`.

I eksemplet tilgår vi elementerne i de indre arrays direkte. Alternativt kunne vi have brugt endnu en løkke til at udskrive værdierne i de indre arrays en af gangen.