

Een Introductie tot PHP

Tom Van Laerhoven

Expertisecentrum Digitale Media 2001
Limburgs Universitair Centrum

1 Inleiding

Deze tekst vormt een introductie tot PHP¹ en behandelt slechts de absolute basis van de taal. Het doel is de lezer een idee te geven van de mogelijkheden en hem vertrouwd te maken met de syntax. Voor een meer gedetailleerde beschrijving van PHP verwijzen we naar de literatuur, en meer bepaald naar het lijstje achteraan dit document.

1.1 Wat is PHP ?

PHP, wat staat voor “*PHP: Hypertext Preprocessor*”, is een *server-side, HTML-embedded* scripting taal. Server-side houdt in dat de code uitgevoerd wordt op de server, dit in tegenstelling tot bijvoorbeeld client-side Javascript dat bij de client zelf wordt uitgevoerd. Het resultaat is dat de gebruiker, de client, het resultaat ontvangt van het uitgevoerde script, zonder enig benul te hebben van de onderliggende code.

De syntax is grotendeels overgenomen uit talen zoals C, Java en Perl maar bevat ook enkele unieke kenmerken. Het doel van PHP is web designers de mogelijkheid te geven om snel dynamische pagina's te genereren.

Het leidende bedrijf achter PHP is *Zend*. Zij hebben ook de Zend engine geschreven, de scripting engine van PHP.

1.2 De mogelijkheden van PHP

Een PHP script kan zowat evenveel als elk ander CGI script, zoals het verzamelen van *form data*, dynamische webpagina's aanmaken, of het versturen of ontvangen van cookies.

Misschien wel de grootste kracht van PHP is de ondersteuning van een heel aantal databases. Momenteel worden een twintigtal verschillende databases ondersteund, waaronder de meest bekende zoals Oracle, PostgreSQL, ODBC, Informix en MySQL.

PHP ondersteunt ook communicatie met andere services, gebruik makend van protocollen als IMAP, SNMP, NNTP, POP3, HTTP, enz.

Achteraan dit document werd een lijst opgenomen met een aantal belangrijke features van PHP.

2 De syntax

PHP is een HTML-embedded taal. Er moet dus op de een of andere manier aan een HTML-parser duidelijk gemaakt worden waar de HTML code ophoudt en de PHP code begint. Dit kan op vier verschillende manieren, waarvan de eerste de voorkeur heeft omdat ze geen extra PHP configuratie-instellingen vereist.

¹In dit document wordt steeds het gebruik van PHP v4 verondersteld.

Listing 1: PHP embedded code.

```
<?php echo("Hello world !"); ?>
<? echo("Hello world!"); ?>
<script language="php"> echo("Hello world!"); </script>
<% echo("Hello world!"); %>
```

2.1 Commentaar

PHP ondersteunt C, C++ en Unix shell-achtige commentaarblokken. Multi-line commentaar wordt omringd door een `/* */` constructie. Geneste commentaar wordt niet ondersteund. Een enkele lijn met commentaar wordt aangegeven door er een `'#'` of een `'//'` voor te plaatsen, en eindigt zodra de volgende begint of zodra de PHP-sectie wordt afgesloten.

Listing 2: Commentaar in PHP

```
<h1>This is an <?php echo "simple"; # Voorbeeld ?> example. </h1>
<h1>This is an <?php echo "simple"; // Voorbeeld ?> example. </h1>

<?php
  /*
    echo "Een test"; /* Deze commentaar veroorzaakt een probleem */
  */
?>
```

2.2 Types

De volgende types worden door PHP ondersteund:

- integer, de grootte is platform-afhankelijk (meestal 32 bits)
- floating-point, de grootte is platform-afhankelijk (meestal 64 bits)
- string
- array
- object

Het type van de variabele (zie sectie ??) wordt meestal niet aangegeven door de programmeur maar wordt *at runtime* door PHP zelf bepaald. Het is wel mogelijk een variabele te casten naar een ander type. Enkele eenvoudige voorbeelden:

Listing 3: Numerieke types in PHP.

```
$a = 1234; # Een decimaal getal
$a = 0123; # Een octaal getal
$a = 0x12; # Een hexadecimaal getal
$a = 1.2e3; # Een floating point
```

2.2.1 Strings

Er zijn een drietal manieren om een *string*-type te definiëren. De eerste maakt gebruik van de *dubbele-quote delimiters* (`"`). Binnenin de quotes is het mogelijk om speciale escape-karakters te gebruiken zoals in C en C++, voorafgegaan door een backslash (`\`). Ook mag men variabelen gebruiken binnenin de string waarvan de waarde in de evaluatie van de string zal voorkomen.

Listing 4: Het gebruik van dubbele-quote delimiters.

```
echo "Myname is : \t $MijnNaam \n";
```

Een andere manier gebruikt de *enkel-quote delimiters* ('). Het verschil met de vorige methode is dat men hierin enkel de "\\" en "\'" mag gebruiken, en dat de waarden van de variabelen *niet* voorkomen in de evaluatie van de string.

Listing 5: Het gebruik van enkel-quote delimiters.

```
echo 'Simple test, \'bla\' ';
```

De derde methode plaatst een *doc-syntax operator* ("<<<") voor de string, gevolgd door een identifier die ook gebruikt wordt om de string af te sluiten. De sluitende identifier *moet* als eerste op de lijn staan.

Listing 6: De doc-syntax operator.

```
echo <<<EOT
  My name is "$name".
  This should print a capital 'A': \x41
EOT;
```

Strings kunnen *geconcateneerd* worden met de dot-operator (.), *niet* met de +-operator. Ook kan men strings behandelen als een getal-geïndexeerde array, zodat toegang tot de aparte karakters mogelijk is.

Listing 7: Concatenatie van strings.

```
$str = "This is a string";
$str = $str . " with some more text";
$str .= " and a newline at the end.\n";
$first = $str[0];
$last = $str[strlen($str)-1];
```

Tenslotte kunnen we strings ook *casten* naar een integer of een floating-point. Indien de string een van de karakters ".", "e" of "E" bevat zal ze als floating-point geëvalueerd worden, in het andere geval zal de evaluatie een integer opleveren, bepaald door het eerste numerieke gedeelte van de string. Foute evaluaties leveren de waarde 0 op.

Listing 8: Automatische casting van strings.

```
$foo = 1 + "10.5"; // $foo is een floating-point (11.5)
$foo = 1 + "bob3"; // $foo is een integer (1)
```

2.2.2 Arrays

Het interessante aan arrays in PHP is dat ze zowel als *associatieve arrays* (er is een associatieve relatie tussen het element en zijn index) als *vectors* (elementen worden aangeduid met een scalaire index) werken. We kunnen dus zowel strings als integers gebruiken als indices:

Listing 9: Voorbeelden van arrays in PHP.

```
$a[0] = "abc";  
$a[1] = "def";  
$b["foo"] = 13;
```

Een array fungeert dus als een dynamische array. Om een waarde aan het einde van de array toe te voegen volstaat het de index weg te laten:

Listing 10: Een dynamische array, cfr. het vector-type in C++.

```
$a[] = "hello"; // $a[2] == "hello"  
$a[] = "world"; // $a[3] == "world"
```

PHP levert een aantal functies om arrays te sorteren, zoals verschillende soorten sort-functies (*asort()*, *arsort()*, *ksort()*, *rsort()*, *sort()*, *uasort()*, *usort()*, *uksort()*), een functie die het aantal elementen uit een array telt (*count()*), en enkele functies die het mogelijk maken een array te doorlopen (*prev()*, *next()* en *each()*).

Multi-dimensionale arrays volgen de C en C++ syntax, met als extra dat ze ook associatief zijn zoals de één-dimensionale arrays. Het toekennen gebeurt zoals voordien, of alternatief met de *array()* of *list()* functie.

Listing 11: Meer-dimensionale arrays.

```
$a[1][0] = $f; # Twee dimensionaal  
$a["foo"][2] = $f; # Mix numerische en associatieve indices  
$a[3]["bar"] = $f; # Mix numerische en associatieve indices  
$a["foo"][4]["bar"][0] = $f; # Vier dimensionaal  
  
$a = array("apple" => array( "color" => "red",  
                             "taste" => "sweet",  
                             "shape" => "round" ),  
          "orange" => array( "color" => "orange",  
                             "taste" => "tart",  
                             "shape" => "round" ),  
          "banana" => array( "color" => "yellow",  
                             "taste" => "paste-y",  
                             "shape" => "banana-shaped" ) );
```

Een vervelend detail is dat multi-dimensionale arrays niet gerefereerd kunnen worden in strings. Dit kan omzeild worden door ofwel gekrulde haakjes rond de array reference te plaatsen of door concatenatie te gebruiken:

Listing 12: Meer-dimensionale arrays als variabelen in een string.

```
$a[3]['bar'] = 'Bob';  
echo "Dit werkt niet: $a[3][bar]";  
echo "Dit wel: " . $a[3][bar];  
echo "Dit ook: {$a[3][bar]}";
```

2.3 Variabelen

Variabelen in PHP zijn *case-sensitive* en beginnen steeds met een dollar teken. Er zijn twee mogelijkheden om waarden toe te kennen aan variabelen: *assign by value* en *assign by reference*.

Bij *assign by value* wordt bij het toekennen van een expressie aan een variabele de waarde van de oorspronkelijke expressie gekopieerd in de doelvariabele. Indien een variabele zijn waarde krijgt *by reference*, wordt deze nieuwe variabele slechts een verwijzing naar de originele variabele. Dit komt overeen met *references* in C en C++, zoals ook de syntax waarbij een ampersand (&) voor de variabele wordt geplaatst.

Listing 13: Assign by reference en assigne by value.

```
$foo = 'Bob';           // Ken de waarde 'Bob' toe aan $foo.
$bar = &$foo;          // $bar is een reference naar $foo.
$bar = "My name is $bar"; // Wijzig $bar ...
echo $foo;             // $foo werd ook gewijzigd.
echo $bar;
```

PHP voorziet een aanzienlijk aantal *voorgedefinieerde variabelen*, afhankelijk van welke webserver er wordt gebruikt, die informatie geven over de webserver en de connectie. Een lijst van al deze variabelen wordt gegeven door de functie `phpinfo()`. Ook alle *environment variabelen* van het systeem worden beschikbaar gesteld.

De *scope* van variabelen verschilt enigzins van variabelen in C en C++. In een functie wordt een lokale scope gebruikt waarin globale variabelen niet meteen voorhande zijn. Deze moeten binnen de functie expliciet als globale variabele aangeduid worden. Dit kan op twee manieren, ofwel door het keyword *global*, ofwel door een speciaal PHP-gedefinieerde array:

Listing 14: De scope van variabelen in functies.

```
$a = 1;
$b = 2;

function sum() {
    echo $a; // Geen output, $a is lokaal gedefinieerd zonder waarde.
}

function sum2() {
    global $a, $b; // 1ste methode: keyword global.
    $b = $a + $b;

    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"]; // 2de methode.
}
```

PHP kent ook *static variabelen*. Ze bestaat enkel in de lokale functie scope, maar bewaart zijn waarde bij het verlaten van de scope, vergelijkbaar met C en C++.

Een interessant concept in PHP zijn de *variabele variabelen*. Dit zijn variabelen waarvan de naam ook variabele is.

Listing 15: Variabele variabelen.

```
$a = 'hello';
$$a = 'world'; // Een variabele met naam $hello en waarde 'world'.
echo "$a ${$a}"; // Beide hetzelfde resultaat.
echo "$a $hello";
```

2.4 Constanten

Een aantal voorgedefinieerd constanten, zoals `__LINE__`, `__FILE__` en `PHP_VERSION` worden voorzien, alsook een mechanisme om zelf constanten te creëren via de `define()` functie.

Listing 16: Het definiëren van constanten in PHP.

```
define("CTTEKST", "Hello World");

function report_error($file, $line, $message) {
    echo "Error in $file, op lijn $line: $message.\n";
}

report_error(__FILE__, __LINE__, "Er ging iets verkeerd !");
```

2.5 Operatoren

De syntax van operatoren in PHP lijkt sterk op die van C en C++. We zullen dan ook vooral diegene benadrukken die verschillend zijn of ontbreken in de syntax van C of C++.

Voorbeeld	Naam	Resultaat
<code>\$a & \$b</code>	Bitwise And	Zie C
<code>\$a == \$b</code>	Equal	Zie C
<code>\$a === \$b</code>	Identical	True als \$a en \$b gelijk zijn en hetzelfde type hebben.
<code>\$a !== \$b</code>	Not identical	True als \$a en \$b verschillen of hun type verschilt.
<code>(e1) ? (e2):(e3);</code>	Condit. o'tor	Zie C
<code>++\$a</code>	Pre-increment	Zie C
<code>\$a or \$b</code>	Logische And	Zie C

De *execution operator*² zorgt ervoor dat de string binnen twee backticks als shell commando wordt uitgevoerd. Let op, dit zijn geen single quotes !

Listing 17: De execution operator.

```
$output = `ls -al`;
echo "<PRE>$output</PRE>";
```

2.6 Controle structuren

De volgende controle structuren worden ondersteund en hebben dezelfde syntax en betekenis als in C en C++.

- if, else
- while, do...while
- for
- break, continue
- switch

Enkele nieuwe structuren werden aan de syntax toegevoegd:

- **elseif**

²In *safe mode* is deze operator niet beschikbaar

- **if(expr): ... else: ... endif;**
- **while(expr): ... endwhile;**
- **for(e1; e2; e3): ... endfor;**
- **foreach(array_expr as \$value) ...**
- **foreach(array_expr as \$key=> \$value) ...**
- **switch(expr): case ... endswitch;**

De *foreach* syntax lijkt op die van de taal perl en maakt het mogelijk om een array te doorlopen. De eerste methode kent per iteratie een element uit de array toe aan *\$value*. De tweede methode is identiek maar hier wordt per iteratie ook de waarde van index (de key) toegekend aan *\$key*. Merk op dat elke *foreach* constructie kan omgezet worden naar een structuur die een *while* lus gebruikt.

De functie *reset()* wordt bij de *foreach* syntax gebruikt om de interne array pointer te initialiseren zodat het naar het eerste element verwijst. Zodra een nieuwe *foreach* loop wordt gestart, wordt deze functie automatisch aangeroepen.

Net zoals een *#include* statement in C, bestaan er in PHP enkele statements die code kunnen importeren.

De functie *require()* vervangt zichzelf door het vernoemde bestand, maar gaat hierdoor uit PHP mode en terug in HTML mode in het begin van het bestand, en omgekeerd aan het einde ervan. Code in dit bestand moet dus ook met PHP tags worden omringd.

Er bestaat ook een *include()* statement dat bijna identiek is aan het *require()* statement met het verschil dat een *require()* statement *altijd* wordt ingelezen, ook indien de lijn waarop het staat nooit wordt uitgevoerd. In deze situatie wordt de code zelf echter niet uitgevoerd. Dit betekent dus dat een *require()* statement binnen een lus geen effect heeft, hier moet men een *include()* gebruiken.

Additioneel worden ook de functies *require_once()* en *include_once()* gebruikt om aan te geven dat de code in het bestand slechts éénmalig moet worden toegevoegd. Dit verhindert anomalieën met variabelewaarden of functienamen.

De *geinclude()*de of *gerequire()*de code erft de scope van de lijn waarop het statement werd uitgevoerd.

2.7 Functies

De syntax van een functie lijkt sterk op die in andere talen. Eender welke geldige PHP code mag binnenin een functie voorkomen, zelfs klassedefinities ! Het volgende voorbeeld toont enkele van de mogelijkheden, zoals het doorgeven van arrays en references als parameter en het gebruik van default waarden.

Listing 18: Een voorbeeld van een functie met verschillende soorten parameters.

```
function mijn_functie( $p_var = 'blah', $p_array, &$p_ref) {
    echo "$p_array[0] + $p_array[1] =", $p_array[0]+$p_array[1];
    $p_ref .= ' met iets erachter';

    return $p_var * $p_var;
}
```

Het is zelfs mogelijk om een variabele als reference door te geven aan een functie terwijl deze functie de parameter niet als reference definieert.

Listing 19: Een parameter als reference doorgeven in een functie.

```
function foo($str) {
    $str .= ' met iets erbij.';
}
$str = 'Mijn string';
foo(&$str);    // $str wordt nu 'Mijn string met iets erbij.'
```

Drie extra functies voorzien de programmeur van meta-data over een functie: *func_num_args()*, *func_get_arg()* en *func_get_args()*. Deze functies maken het ook mogelijk om een functie te ontwerpen met een variabel aantal parameters, vergelijkbaar met de *ellipsis* in C++.

De volgende methode laat toe meerdere waardes te laten teruggeven door een functie, gebruik makend van een array die wordt opgebouwd door de functie *array()*:

Listing 20: Het teruggeven van meerdere waarden door een functie.

```
function small_numbers() {
    return array(0, 1, 2);
}
list($zero, $one, $two) = small_numbers();
```

Tenslotte ondersteunt PHP *variabele functies*, vergelijkbaar met de variabele variabelen uit sectie ??.

Listing 21: Variabele functies.

```
functie foo() {
    echo "In foo()\n";
}
$func = 'foo';
$func();    // foo() wordt uitgevoerd.
```

2.8 Objecten en klassen

Net zoals in C++ vormt een klasse in PHP een blueprint van een object.

Listing 22: Een voorbeeld van een klassedefinitie.

```
class Cart {
    var $items; // Items in our shopping cart

    function Cart() {
        $this->add_item("03", 0);
    }

    function add_item($artnr, $num) {
        $this->items[$artnr] += $num;
    }
}

$cart = new Cart;
$cart->add_item("10", 1);
```

Het is mogelijk om klassen af te leiden met het *extends* keyword. Multiple inheritance wordt echter niet ondersteund. Bij het creëren van een object wordt de constructor van de basisklasse *niet* expliciet aangeroepen.

Listing 23: Een voorbeeld van inheritance.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

3 PHP Features

3.1 Error handling

PHP bevat een aantal waarden die worden gebruikt om foutmeldingen te rapporteren. Deze waarden vormen tezamen een bitmask die ondervraagd kan worden met de bitwise operatoren.

De functie *set_error_handler()* laat de gebruiker toe een eigen error callback-functie te definiëren. Zodra ergens in de code de functie *trigger_error()* wordt aangeroepen, wordt deze callback-functie geactiveerd.

Een speciale operator, de *error-control operator* '@', zorgt ervoor dat foutmeldingen die het gevolg zijn van parse errors afgezet worden. Dit doet men door deze operator voor een expressie te plaatsen.

3.2 Het manipuleren van beelden

PHP is niet beperkt tot het genereren van HTML. Men kan ook beeldbestanden creëren en manipuleren, waaronder gif, png, jpg, wbmp en xpm bestanden. De output-streams kunnen direct naar de browser gestuurd worden.

3.3 En verder ...

- HTTP authentication
- Cookies
- Handling file uploads
- Het gebruik van remote files
- ...

4 PHP vs. alternatieven

De twee meest gebruikte alternatieven voor PHP zijn *ColdFusion* en *ASP*.

Bij het ontwerpen van een data-driven website ligt de keuze niet echt voor de hand, de drie engines hebben immers dezelfde functionaliteiten. Een keuze zal dus gemaakt worden op basis van bestaande hardware en software investeringen, ofwel op basis van een voorkeur voor een bepaalde engine.

Er is al veel gediscussieerd over welke van de drie het beste zou zijn. Echte conclusies kunnen uit dit soort discussies zelden getrokken worden. Wij beperken ons hier tot een korte opsomming van enkele objectieve observaties.

- PHP kan veel, maar vereist (gratis) 3rd party addons.
- PHP is sneller omdat er geen communicatie is tussen COM-objecten (PHP is immers niet COM-gebaseerd).
- PHP, ASP en CF zijn beschikbaar op zowat alle platformen.
- De ASP voor niet-Microsoft platformen is commercieel.
- PHP is open source, gratis.
- ASP heeft veel minder built-in functionaliteiten.

5 Links

- <http://www.php.net>
- <http://www.phpwizard.net>
- <http://www.phpbuilder.com>
- <http://conf.php.net>
- <http://php.pagina.nl>
- <http://php.resourceindex.com>
- <http://www.hotscripts.com/PHP/>
- <http://www.zend.com>
- <http://www.cyberglitz.com/>
- Newsgroups: php.doc, php.general, alt.comp.lang.php, ...