

Web Expert 2

Deel 3: PHP

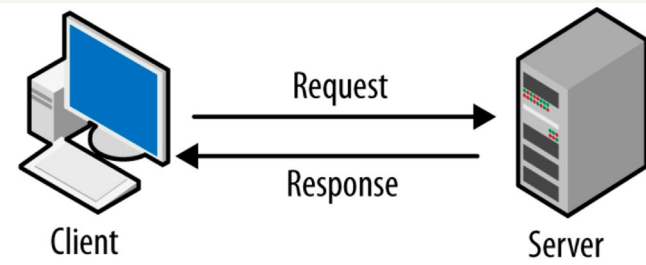


PHP

PHP is een scripttaal die gebouwd is in functie van web development. Initieel stond PHP voor Personal Home Page, maar doorheen de evolutie die de taal heeft meegemaakt, staat PHP nu voor PHP: Hypertext Preprocessor.

PHP wordt meestal verwerkt op een web server om dynamische webpagina's weer te geven. We gaan hier opnieuw terug naar de communicatieflow die in Web Expert 1 ook al aan bod is gekomen:

De browser stuurt een request naar de server. Op de server staat een PHP interpreter die het programma zal uitvoeren en het resultaat als response terug naar de browser zal sturen.



We gebruiken deze code als voorbeeld.

Je kan zien dat het bestand niet alleen HTML code bevat, maar ook PHP code.

Deze code staat in een PHP tag:

```
<?php ... ?>
```

Bij het uitvoeren van de code zal de PHP interpreter elke PHP tag vervangen door de output die in de tag wordt gegenereerd.

```
<html lang="en">
<head>
  <meta charset="UTF8">
  <title>Flow</title>
</head>
<body>
<?php
    $text = 'Communicatie voorbeeld';
    print( "Hello!! $text\n" );
?>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF8">
    <title>Flow</title>
  </head>
  <body> Hello Communicatie voorbeeld </body>
</html>
```

Variabelen

In PHP wordt een variabele aangeduid door een dollarteken, gevolgd door een aantal geldige symbolen. Bijvoorbeeld **\$text**.

De variabelen zijn case-sensitive. `$text` en `$Text` zijn dus 2 verschillende variabelen.

PHP is een flexibele programmeertaal. De variabelen moeten niet gedeclareerd worden en elke variabele kan waarden van eender welk type bevatten. Het type wordt dan bepaald door de context.

Sinds PHP 7.4 is het echter wel mogelijk om in bepaalde omstandigheden een variabele te declareren. Bijvoorbeeld voor klasse variabelen, argumenten aan functies,

Om dit aan te tonen vertrekken we van volgend voorbeeld:

```
<?php
$a = 1;
echo('a heeft waarde ' . $a . ' en type ' . gettype($a) . "\n");
var_dump($a);
```

De code kunnen we uitvoeren door in de terminal naar de map te navigeren en het volgende commando in te geven:
php <naam van het bestand>

Bij het uitvoeren geeft dit volgend resultaat.

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\variabelen.php
a heeft waarde 1 en type integer
int(1)
```

Het type vragen we op met de functie `gettype()` en de volledige informatie van de variabele wordt afgedrukt met de functie `var_dump()`.

Probeer dit nu zelf door `$a` de waarde `"tekst"` te geven en nadien `$a` de waarde `true` te geven.

Cast operaties

Het is ook mogelijk omzettingen tussen verschillende types te maken via cast operaties.

```
<?php

$a = "";
print ("\$a is van het type " . gettype($a) . "\n");
var_dump($a);

$b = (boolean)$a;
if ($b) {
    print ("true \n");
} else {
    print ("false \n");
}

print ("\$b is van het type " . gettype($b) . "\n");
var_dump($b);
```

In het voorbeeld krijgt de variabele `$a` een lege string als waarde.

De inhoud van de variabele `$a` wordt vervolgens via een cast operatie omgezet naar het type `boolean` en het resultaat wordt in de variabele `$b` geplaatst.

Dit geeft volgend resultaat:

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\cast.php
$a is van het type string
string(0) ""
false
$b is van het type boolean
bool(false)
```

Hieruit kunnen we concluderen dat een lege `string` wordt omgezet naar de boolean `false`.

Andere waardes die resulteren in `false`:

- De integer `0`
- De double `0.0`
- Een `string` met de waarde `"0"`
- `null`

Alle andere waardes van het type `integer`, `double` of `string` worden naar `true` gecast.

Casten naar een integer

Dit gebeurt via de operaties `(integer)` of `(int)` en hierbij gelden de volgende regels:

- De `boolean true` wordt gecast naar 1, `false` naar 0.
- Een `double` wordt naar beneden afgerond.
- Een `string` die niet begint met een cijfer wordt omgezet naar 0. Een `string` die wel begint met een aantal cijfers wordt omgezet naar het overeenkomende getal. Spaties in het begin van de string worden hierbij genegeerd. Ook worden symbolen na de reeks cijfers niet in rekening gebracht.
- `null` wordt 0.

Casten naar een double of float

Deze operatie verloopt analoog aan de cast operatie naar een integer.

Casten naar een string

Dit gebeurt op basis van de cast operatie `(string)`.

- Van numerieke waarden wordt een `string` representatie gegeven.
- De boolean `true` wordt omgezet naar `"1"`, de boolean `false` naar een lege `string`.
- Ook `null` wordt omgezet naar een lege `string`.

Operatoren

In PHP kan je net als in alle andere programmeertalen, wiskundige operatoren gebruiken.

Deze operaties verwachten altijd twee numerieke argumenten. Het resultaat is dan ook steeds numeriek.

Wanneer de bewerkingen `+`, `-` en `*` één van de argumenten een `float` is (of een `string` die kan omgezet worden naar een `double`), dan worden beide beschouwd als een `float` en zal het resultaat een `float` zijn. Is die niet het geval, dan zullen ze geëvalueerd worden als een `integer` en zal het resultaat een `integer` zijn.

```
var_dump( value: 1 + 1.22 );
```



```
float(2.2199999999999998)
```

Bij een deling worden beide argumenten als `float` beschouwd en is het resultaat een `float`.

Bij een modulo (%) bewerking worden de argumenten steeds als `integer` beschouwd en is het resultaat een `integer`.

De belangrijkste tekstoperatie is de concatenatie waarmee 2 strings met elkaar worden verbonden. Hiervoor kan `.` gebruikt worden.

De argumenten links en rechts van deze operatie worden beide als string beschouwd en indien nodig naar string gecast.

```
var_dump( value: ' - ' . true . ' - ' . false . ' - ' );
```



```
string(10) " - 1 - - "
```

Met de vergelijkingsoperatoren kunnen twee waarden vergeleken worden. Het resultaat van zo een operatie is altijd een `boolean`, naargelang de waarden al dan niet voldoen aan de voorwaarde.

<code>==</code>	Gelijk in waarde
<code>===</code>	Gelijk in waarde en van hetzelfde datatype
<code>!=</code>	Verschillend in waarde
<code>!==</code>	Verschillend in waarde of datatype
<code><</code>	Kleiner dan
<code>></code>	Groter dan
<code><=</code>	Kleiner dan of gelijk aan
<code>>=</code>	Groter dan of gelijk aan

```
var_dump( value: 1 == '1');  
var_dump( value: 1 === '1');
```

```
bool(true)  
bool(false)
```

Net zoals in andere programmeertalen kunnen we ook logische operatoren gebruiken: && (and), || (or)

En is het mogelijk om operatoren te gebruiken die een waarde toekennen op basis van een conditie:

- Ternary: ?:

```
$x = (1 == 2) ? 5 : 3;  
var_dump($x);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\operatoren.php  
int(3)
```

- Null coalescing: ?? : zal de waarde van de eerste expressie terug geven indien deze niet null is, anders zal de waarde van de tweede expressie worden terug gegeven.

```
$x = (1 == 2) ?? ("Andere expressie");  
var_dump($x);  
$x = (null) ?? ("Andere expressie");  
var_dump($x);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\operatoren.php  
bool(false)  
string(16) "Andere expressie"
```

Controle structuren

Ook de controle structuren zijn te vergelijken met die van de andere programmeertalen.

- If - else
- While
- For

```
$a = 0;

//If - else
if ($a < 1) {
    print("\$a is kleiner dan 1\n");
} else {
    print("\$a is groter dan 1\n");
}

//While
while ($a < 10) {
    print($a);
    $a++;
}
print("\n");

//for
for ($i = 0; $i < 10; $i++) {
    print($i);
}
print("\n");
```

Arrays

In PHP is een array een datatype waarin een aantal waarden (values) bewaard kunnen worden. Bij elke waarde in de array is er een sleutel (key) voorzien waarmee de waarde opgevraagd kan worden.

Een array kan aangemaakt worden via de functie `array()`:

```
$a = array(element_1, element_2, ..., element_N);
```

Of op de volgende manier: `$a = [element_1, element_2, ..., element_N]`

Waarbij:

```
element_i = value_i
```

of

```
element_i = key_i => value_i
```

In bovenstaande definitie wordt de lijst `$a`, bestaande uit `N` elementen, aangemaakt. Per element wordt ofwel enkel de waarde gespecificeerd ofwel wordt de combinatie van de sleutel en de waarde opgegeven.

De waarden in een `array` moeten niet allemaal van hetzelfde type zijn.

De sleutel moet van het datatype `integer` of `string` zijn. Wanneer er geen key wordt gespecificeerd bij het aanroepen van de functie `array()`, dan wordt de eerstvolgende beschikbare integer genomen als key.

```
<?php
$a = array(1, 2, 3, 4);
$b = array(1, 2.1, true, "ja");
$c = array(1 => 12, "Juist" => true);
$d = array (2 => 1, 10);
echo '$a = array(1,2,3,4)';
var_dump($a);

echo '$b = array(1,2.1,true,"ja")';
var_dump($b);

echo '$c = array(1=> 12, "Juist" => true)';
var_dump($c);

echo '$d = array (2 => 1, 10)';
var_dump($d);
```

```
$a = array(1,2,3,4)
array(4) {
    [0]=>
    int(1)
    [1]=>
    int(2)
    [2]=>
    int(3)
    [3]=>
    int(4)
}
```

```
$b = array(1,2.1,true,'ja')
array(4) {
    [0]=>
    int(1)
    [1]=>
    float(2.1)
    [2]=>
    bool(true)
    [3]=>
    string(2) "ja"
}
```

```
$c = array(1=> 12, 'Juist' => true)
array(2) {
    [1]=>
    int(12)
    ["Juist"]=>
    bool(true)
}
```

```
$d = array (2 => 1, 10)
array(2) {
    [2]=>
    int(1)
    [3]=>
    int(10)
}
```


Na de creatie van een `array` kunnen de waarden opgehaald en gewijzigd worden aan de hand van hun sleutels. Een waarde uit de rij wordt geselecteerd via vierkante haken:

`$rij[key_i]`: Selecteer de waarde die overeenkomt met `key_i`

```
$a = array(0, 2, 3, 4);
echo "$a[0]\n";
$a[0] = "ja";
var_dump($a);

echo "\n";

$b = array("een" => 1, "twee" => 3 );
echo $b["een"] . "\n";
echo "$b[een]\n";
$b["twee"] = 2;
var_dump($b);
echo "\n";
```

```
0
array(4) {
  [0]=>
  string(2) "ja"
  [1]=>
  int(2)
  [2]=>
  int(3)
  [3]=>
  int(4)
}
```

```
1
1
array(2) {
  ["een"]=>
  int(1)
  ["twee"]=>
  int(2)
}
```

We kunnen na de creatie van een lijst, nog steeds elementen toevoegen

```
$a = array( 1, 2, 3, 4 );  
$a[7] = 11;  
var_dump( $a );  
  
echo "\n";  
  
$b = array( "een" => 1, "twee" => 3 );  
$b["drie"] = 3;  
var_dump( $b );  
echo "\n";
```

```
array(5) {  
  [0]=>  
  int(1)  
  [1]=>  
  int(2)  
  [2]=>  
  int(3)  
  [3]=>  
  int(4)  
  [7]=>  
  int(11)  
}
```

```
array(3) {  
  ["een"]=>  
  int(1)  
  ["twee"]=>  
  int(3)  
  ["drie"]=>  
  int(3)  
}
```

Met lege vierkante haken, kunnen we ook een waarde toevoegen: `$rij[] = value_i`

De waarde `value_i` wordt in de rij geplaatst met als key de eerstvolgende vrije index.

Elementen verwijderen gebeurt aan de hand van de `unset()` functie:

```
$d = array( 1, "a" => 2, "b" );  
unset( $d["a"] );  
var_dump( $d );
```

```
array(2) {  
    [0]=>  
    int(1)  
    [1]=>  
    string(1) "b"  
}
```

Andere nuttige functies die toegepast kunnen worden op een array zijn de volgende:

<code>count(\$a)</code>	Geef de lengte van de array \$a
<code>array_keys(\$a)</code>	Geef een lijst met alle keys van de array \$a
<code>array_keys(\$a, \$val)</code>	Geef een lijst met de keys uit de array \$a die overeenkomen met de waarde van \$val.
<code>array_values(\$a)</code>	Geef een lijst met alle waarden van de array \$a
<code>sort(\$a)</code>	Sorteer de array \$a
<code>shuffle(\$a)</code>	Schud de array \$a
<code>min(\$a)</code>	Geef het minimum van de array \$a
<code>max(\$a)</code>	Geef het maximum van de array \$a

Foreach

Foreach is een handige structuur die gebruikt kan worden om door de waarden (en de sleutels) van een array te gaan.

In volgend voorbeeld wordt elke waarde van de array `$rij` in de variabale `$v` gekopieerd.

```
foreach($rij as $v) { ... }
```

```
<?php
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");
foreach ($a as $v) {
    echo "value: $v \n";
}
```

```
value: ma
value: di
value: wo
value: ma
```

Via een tweede vorm van de foreach lus is zowel de waarde als de sleutel van elk element uit de rij beschikbaar in de sequentie:

```
foreach($rij as $k => $v) { ... }
```

```
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");  
foreach ($a as $k=>$v) {  
    echo "key: $k, value: $v \n";  
}
```

```
key: 1, value: ma  
key: 2, value: di  
key: 3, value: wo  
key: 4, value: ma
```

Funcities

Funcities bevatten code die uitgevoerd wordt wanneer de functie wordt aangeroepen. Bij het aanroepen kunnen argumenten worden meegegeven. Een waarde kan terug gegeven worden aan de hand van het `return` keyword.

```
<?php
function som($getal1, $getal2)
{
    return $getal1 + $getal2;
}

function printResultaat($resultaat)
{
    print ($resultaat);
}

$resultaat = som( getal1: 1, getal2: 2);

printResultaat($resultaat);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\funcities.php
3
```

Je kan aan argument in een functie ook een standaard waarde meegeven. Hierdoor kan de functie aangeroepen worden wonder of met het argument dat een standaard waarde heeft.

```
function som($getal1, $getal2, $getal3 = 0)
{
    return $getal1 + $getal2 + $getal3;
}

$resultaat = som( getal1: 1, getal2: 2);
print ($resultaat);

$resultaat = som( getal1: 1, getal2: 2, getal3: 3);
print ($resultaat);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\functies.php
3
6
```

In het vorige voorbeeld werd een manier aangehaald om een functie te kunnen hergebruiken. Een ander manier om dit te doen is een functie aanmaken zonder argumenten en gebruik te maken van de functie **func_get_args()**. De functie kan nu met eender welk aantal argumenten aangeroepen worden. Deze argumenten worden opgevraagd en in een array geplaatst.

```
function som()
{
    $som = 0;
    $argumenten = func_get_args();
    foreach ($argumenten as $argument) {
        $som += $argument;
    }
    return $som;
}

$resultaat = som(1, 2);
print ($resultaat);

print("\n");

$resultaat = som(1, 2, 3, 7, 9);
print ($resultaat);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\functies.php
3
22
```


Als alternatief kunnen we ook de spread operator gebruiken.

```
function som(...$getallen)
{
    $som = 0;
    foreach ($getallen as $getal) {
        $som += $getal;
    }
    return $som;
}

$resultaat = som(...getallen: 1, 2);
print ($resultaat);

print("\n");

$resultaat = som(...getallen: 1, 2, 3, 7, 9);
print ($resultaat);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\functies.php
3
22
```

Sinds PHP 7 kunnen we types meegeven aan de argumenten van een functie en kunnen we aangeven wat het return type van de functie moet zijn.

```
<?php
function printDatum(DateTime $date) : void
{
    print ($date->format( format: 'Ymd H:i:s'));
}

function printRij(array $rij) : void
{
    foreach ($rij as $selement) {
        print($selement . "\n");
    }
}

$date = new DateTime();
printDatum($date);

print("\n");

$rij = [1, 2, 3];
printRij($rij);
```

```
PS C:\Users\20003967\Documents\Werkmap\php_voorbeelden> php .\types.php
20220213 20:04:59
1
2
3
```

Voorgedefinieerde functies

In PHP zijn er heel wat voorgedefinieerde functies beschikbaar. We overlopen er een aantal.

String functies

<code>strlen(\$a)</code>	Geef de lengte van de string \$a
<code>strpos(\$a, \$b)</code>	Geef de eerste positie van een zoekstring \$b in de string \$a. Indien \$b niet gevonden worden wordt is het resultaat false.
<code>substr(\$a, \$i)</code>	Geef de substring van de string \$a, beginnend vanaf de positie \$i.
<code>strtolower(\$a)</code>	Geef de string \$a, omgezet naar kleine letters, terug.
<code>strtoupper(\$a)</code>	Geef de string \$a, omgezet naar hoofdletters, terug.
<code>trim(\$a)</code>	Geef de string \$a terug zonder spaties voor- en achteraan.

Functies voor variabelen

<code>define("PI", 3,1415)</code>	Definieer een constante.
<code>unset(\$a)</code>	Verwijder de variabele \$a.
<code>isset(\$a)</code>	Gaat na of de variabele \$a bestaat.
<code>is_string(\$a)</code>	Gaat na of de variabele \$a een string is.
...	
<code>trim(\$a)</code>	Geef de string \$a terug zonder spaties voor- en achteraan.

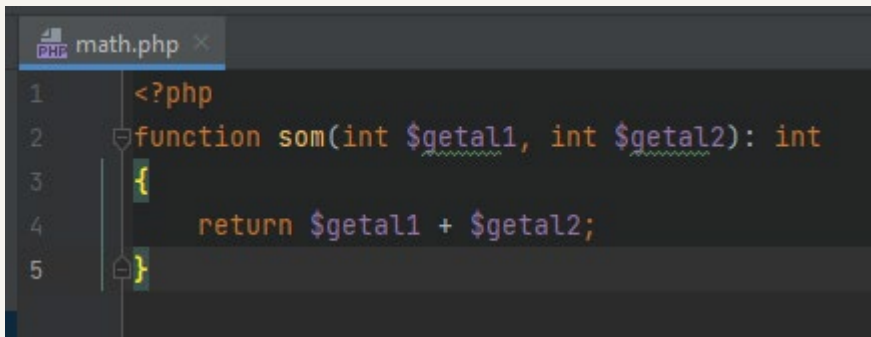
Include en require

Met de functies `include` en `require` wordt een bestand geopend, de PHP code in dit bestand wordt uitgevoerd en het resultaat wordt geplaatst op de positie waar de functie aangeroepen wordt.


Het verschil tussen `include` en `require` ligt in de manier waarop fouten worden behandeld. Indien het te openen bestand niet bestaat, zal bij `include` een warning gegeven worden, terwijl bij `require` er een `fatal error` zal optreden.

`Include_once` en `require_once` zijn vergelijkbaar met `include` en `require`, ze zorgen er enkel voor dat elk bestand maar 1 keer moet geïmporteerd worden. Door middel van deze functies kunnen programma's makkelijk opgesplitst worden in meerdere bestanden.

Traditioneel zal het `autoload` bestand in het hoofdbestand worden ingeladen met een `require_once` opdat de composer dependencies in elk onderliggend bestand beschikbaar zijn.

A screenshot of a code editor window titled 'math.php'. The code defines a function named 'som' that takes two integer parameters, '\$getal1' and '\$getal2', and returns their sum. The function is enclosed in curly braces, and the return statement is indented.

```
1 <?php
2 function som(int $getal1, int $getal2): int
3 {
4     return $getal1 + $getal2;
5 }
```

A screenshot of a code editor window titled 'som.php'. The code uses 'require_once' to include 'math.php', then calls the 'som' function with arguments 1 and 2, and prints the result. The variable '\$resultaat' is used to store the result of the function call.

```
1 <?php
2 require_once('math.php');
3
4 $resultaat = som(1, 2);
5 print ($resultaat);
```