MINICURSUS PHP

PHP Cursus

Deze cursus is om de eerste stappen in de wereld van webdevelopment te zetten. De volgende onderwerpen zullen naar boven komen:

- Wat is PHP?
- Programmeren met PHP
  - Programmastructuur
  - Variabelen
  - Datatypen
    - Integers en floats
    - Strings
    - Arrays
  - Operators
  - Lussen
    - `if`
    - `while`
    - `for`
  - Functies
  - Objecten / OO programmeren

Wat is PHP?

PHP is een programmeertaal die ontwikkeld is om *webapplicaties* mee te kunnen ontwikkelen. Het is een taal met een lange geschiedenis en een aantal grote veranderingen in deze geschiedenis.

Over hoe PHP precies de wereld in is gekomen zal ik het niet hebben, dat kun je vinden op de website van PHP zelf bij hun geschiedenis pagina.

Waar draait PHP op?

Van oorsprong is PHP ontwikkeld om te draaien op *Unix* en *Unix achtigen* (Linux), maar omdat de broncode van PHP voor iedereen in te zien is draait het tegenwoordig op zo'n beetje elk OS.

Wat draait er op PHP?

Teveel om op te noemen. Grote projecten als Wordpress en Drupal, maar ook websites als Tweakers en Facebook.

Wat heb je nodig om met PHP te programmmeren?

De volgende zaken:

- Een computer waar PHP op draait, eventueel met een database server. Dit laatste hoeft echter niet.
- Een editor. Editors zijn er teveel om op te noemen, en ieder heeft zijn/haar eigen voorkeur. Ik gebruik zelf Atom, maar echt uitmaken doet het niet. Gebruik wat je zelf fijn vind.
- Een webbrowser om je code te testen.

Programmeren met PHP

Net zoals elke andere programmeertaal heeft PHP een aantal regels en vaste afspraken. De meeste van deze zijn lastig uitleggen en komen in de cursus vanzelf naar boven. Een aantal zijn echter wél uit te leggen, dus die komen hier voorbij.

Programmastructuur

Elk programma, van het simpelste script tot een systeem als Wordpress, heeft een zogenaamde *flow*, een structuur. Code zomaar ergens neergooien en hopen dat het het doet is een optie, maar niet eentje waar je ver mee gaat komen.

Om een oude traditie in de programmeerwereld aan te houden, hier een demo PHP programma wat niets meer doet dan de woorden `Hallo, wereld` op je scherm laten zien:

```
<?php
print "Hallo, wereld";
```

Zoals je ziet begint dit programma, en elk PHP programma, met `<?php` gevolgd door een enter. Vergeet je dit zal je programma weinig doen, omdat PHP dan niet aangeroepen word. Het enige wat je dan ziet is `print "Hello, world";` in je browser.

Variabelen

Elke programmeertaal kent *variabelen*. Een *variable* is, kort door de bocht, een stuk in het geheugen waarvan jij kunt beslissen wat je erin stopt. Een voorbeeld, met het `Hello world` programma:

```
<?php
# Dit is een variabele hieronder. Oh, en dit is commentaar!
$tekst = "Hallo, wereld";

# Print $tekst naar het scherm
print $tekst;
```

Zoals je kunt zien is er een variabele `$tekst` met als inhoud "Hallo, wereld". Deze variabele word op de laatste regel op het scherm getoond.

Wat heeft dit nu voor zin? In het voorbeeld hierboven is het extra code die niets extras toevoegt hoor ik je denken, en dat klopt. Variabelen komen pas tot hun recht als je gaat werken met lussen als `if`, `for` en `while`, om maar een voorbeeld te geven. Hier zul je later in dit document wat voorbeelden van zien.

Datatypen

Zoals bijna elke andere programmeertaal kent PHP *datatypes*. Wat dit precies zijn zal ik je besparen (dit is te vinden in de PHP Documentatie), weet dat er een aantal zijn, waarvan de belangrijkste die we hier gaan behandelen de volgende zijn:

- `Integers`: gehele getallen, bijvoorbeeld *25* of *65536*
- `Floats`: Kommagetallen, bijvoorbeeld *3.14*
- `Strings`: Tekst, bijvoorbeeld "Hallo Wereld". Je herkent een string aan het feit dat er (dubbele) quotes omheen staan
- `Boolean`: Ja of nee, `true` of `false` (`1` en `0` werken ook)
- `Arrays`: Lijsten met items, wat je in een array stopt maakt niet zoveel uit

Integers en floats

Getallen in PHP heb je in twee varianten. Gehele getallen (7 bijvoorbeeld) en kommagetallen (3.14). Een geheel getal heeft als datatype `int` en een kommagetal heeft als datatype `float`.

Hier een kort voorbeeld met getallen, en meteen wat wiskunde!

```
<?php
# Declareer een variabel met als inhoud het getal 2
# Omdat 2 een geheel getal is is het een int
$getal = 2;

# Wat is $getal * 2?
$uitkomst = $getal * 2;
print $uitkomst;

# Wat is $uitkomst * $getal ?
print $uitkomst * $getal;

# En dan nu floats. Pi leent zich hier goed voor.
# Declareer een variabele met als inhoud Pi (3.1415...)
# Omdat Pi een kommagetal is is het een float
$pi = 3.1415;

# Wat krijg je als je Pi * Pi doet?
print $pi * $pi;

# En Pi * Pi * Pi?
print $pi * $pi * pi;

# En als afsluiting, Pi - $uitkomst van hierboven
print $pi - $uitkomst;
```

Strings

Een string is, behalve een stuk ondergoed, een stuk tekst. Dit heb je in het *Hallo, Wereld* voorbeeld al kunnen zien, maar toch ga ik hier even wat dieper op in.

In elke taal is het handig om de mogelijkheid te hebben om tekst aan je gebruiker(s) te kunnen laten zien. Denk bijvoorbeeld aan (fout)meldingen of om een vraag te stellen. Voorbeeld:

```
<?php
print "Pi is 3.1415 en dan nog een heleboel getallen.";
```

Arrays

Arrays zijn een krachtig hulpmiddel in elke (moderne) programmeertaal. Bijna elke taal heeft ze, of iets wat erop lijkt. In Python worden arrays ook wel `list`s genoemd. Een array is niets meer dan een lijst met daarin variabelen, zoals dit:

```
<?php
# Declareer een array met Minions op de ouderwetse manier
$minions = array('Kevin', 'Stuart', 'Bob');

# Declareer dezelfde array met Minions op de nieuwerwetse manier.
# Let wel op dat dit alleen werkt in PHP >= 5.4
$minions = ['Kevin', 'Stuart', 'Bob'];
```

Je kunt en array gebruiken om (tijdelijk) een aantal dingen op te slaan. Als je nu Stuart uit de `$minions` array wilt halen

doe je dat zo:

```
<?php
# Haal Stuart uit de $minions array
print $minions[1];
```

Valt je iets op? Stuart staat als 2e in de lijst, maar toch pak ik hierboven nummer 1?! Dat is omdat in PHP's arrays beginnen bij tellen vanaf 0. Dus 0 is Kevin, 1 is Stuart en 2 is Bob.

Het ophalen van één waarde uit een array doe je door de teller op te geven tussen blokhaken ([en]) na de array naam (`$minions` in ons geval).

Operators

Net als bij wiskunde kent PHP een aantal (wiskundige) operators. De werking is bij een aantal hetzelfde. Voor de zekerheid hier een lijstje van de meest gebruikte. Mocht je ze allemaal willen zien kun je terecht in de PHP operators handleiding

- `<` : Kleiner dan
- `<=` : Kleiner dan of gelijk aan
- `>` : Groter dan
- `=>` : Groter dan of gelijk aan
- `==` : Is gelijk aan
- `===` : Is gelijk aan en van hetzelfde datatype (*identical* operator)

Lussen

Elke zichzelf respecterende programmeertaal heeft lussen, soms ook wel loops genoemd. Zo ook PHP.

Met een lus kun je een actie een aantal keer herhalen, of kijken of er aan een voorwaarde voldaan is en dan pas iets doen, of over alle elementen in een `array` gaan en iets daarmee doen.

Het voorbeeld hieronder kijkt of de variabele `$naam` gelijk is aan *Kevin*. Als dit zo is word de boodschap *Banana?* getoond. Als `$naam` niet gelijk is aan *Kevin* word de boodschap *Hallo, Wereld!* getoond:

`if` lus (voorbeeld)

```
<?php
# Dit is een "if" lus voorbeeld

# Declareer de variabele $naam en stop er een string in
$naam = "Stuart";

# Kijk of $naam gelijk is aan Kevin, zo ja laat je Banana? zien,
# Zo nee laat je Hallo Wereld zien
#
# Dit is een if (if-else) lus
if($naam == "Kevin")
{
        print "Banana?";
}
else
{
        print "Hallo, Wereld";
}
```

`while` lus (voorbeeld)

Je kunt met lussen ook iets doen zolang er wel (of niet) aan een *voorwaarde* voldaan word. Een voorbeeld:

```
<?php
# while lus voorbeeld

# De huidige leeftijd van onze fictieve persoon
$leeftijd = 18;

# Pensioengeld, word per jaar meer. We beginnen
# met 1000 euro
$pensioen = 1000;

while($leeftijd <= 67)
{
        # Elk jaar komt er 120 euro per maand bij,
        # dus eerst wat rekenewerk om 120 * 12 te doen
        $bedrag = 120 * 12;

        # Tel $bedrag op bij $pensioen
        # Dit had ook zo gekund:
        # $pensioen = $pensioen + $bedrag;
        $pensioen += $bedrag;

        # De $leeftijd met 1 (jaar) ophogen, anders blijven
        # we bezig!
        $leeftijd = $leeftijd + 1;
}

# Eens kijken wat onze fictieve persoon aan pensioen heeft
# met ~40 dienstjaren
print $pensioen;
```

`for` lus (voorbeeld)

Het `while` voorbeeld hierboven kan ook met een `for` lus. De werking is grotendeels hetzelfde, het verschil zit hem in het feit dat je in een `for` lus in één klap de variabele declareert, bepaalt hoelang de lus loopt en hoe de telling moet verlopen.

```
<?php
# while lus voorbeeld

# We beginnen weer met 1000 euro pensioen
$pensioen = 1000;

# En de leeftijd is weer 18, pensioenleeftijd is 67
# '$leeftijd++' is hetzelfde als '$leeftijd = $leeftijd + 1'
for($leeftijd = 18; $leeftijd <= 67; $leeftijd++)
{
        # Weer 120 euro per maand, maal 12
        $bedrag = 128 * 12;

        # Bedrag bij $pensioen optellen
        $pensioen += $bedrag;
}

# Eens kijken hoeveel er in de pensioenpot zit
print $pensioen;
```

Functies

Een functie is een stuk code wat je kunt gebruiken om veel voorkomende handelingen te automatiseren. Je hebt in PHP het onderscheid tussen zelf geschreven functies en functies die vanuit de taal geleverd worden, zoals `if`, `for`, `while` e.a.

Hier een voorbeeld van een zelfgeschreven PHP functie die, afhankelijk van het tijdstip, een begroeting op het scherm print.

```
<?php
# Print een begroeting naar het scherm
# afhankelijk van het tijdstip.

function groet()
{
        $uur = date('H');

        if($uur >= 0 && $uur <= 6)
        {
                $begroeting = 'Goedenacht';
        }
        elseif($uur >= 6 && $uur <= 12)
        {
                $begroeting = 'Goedemorgen';
        }
        elseif($uur >= 12 && $uur <= 18)
        {
                $begroeting = 'Goedemiddag';
        }
        else
        {
                $begroeting = 'Goedeavond';
        }

        return $begroeting;
}

print groet();
```

Soms wil je wat flexibeler zijn in je groeten en een naam mee kunnen geven. Dit zou je kunnen doen door voor elke naam een eigen functie te schrijven, maar dat is wat omslachtig.

Gelukkig is er een andere oplossing: *argumenten*. Je kunt één of meerdere 'opties' (argumenten) aan een functie meegeven en die in de functie gebruiken; bijvoorbeeld een naam van de persoon die je groet. Dat doe je zo:

```
<?php
# Print een gepersonaliseerde begroeting
# naar het scherm afhankelijk van het
# tijdstip

function groet($naam)
{
        $uur = date('H');

        if($uur >= 0 && $uur <= 6)
        {
                $begroeting = 'Goedenacht';
        }
        elseif($uur >= 6 && $uur <= 12)
        {
                $begroeting = 'Goedemorgen';
        }
        elseif($uur >= 12 && $uur <= 18)
        {
                $begroeting = 'Goedemiddag';
        }
        else
```

```
        {
                $begroeting = 'Goedeavond';
        }

        # De " . ' ' . "  is om twee
        # strings aan elkaar te plakken.
        #
        # Dit heet "concatinatie".
        return $begroeting . ' ' . $naam;
}

print groet("Stuart");
```

PHP 7 en functies

Sinds de release van PHP 7 een paar maanden terug zijn er wat dingen veranderd in PHP met betrekking tot functies. Je kunt in PHP 7 nu opgeven wat je functie terug moet geven voor datatype (`string` , `int` , of iets anders) en wat het datatype van een of alle argumenten moet zijn. Voorbeeld, met de `groet` functie hierboven.

```
<?php
# Print een gepersonaliseerde begroeting
# naar het scherm afhankelijk van het
# tijdstip
#
# De functie groet moet nu verplicht een string
# teruggeven, en het argument $naam moet verplicht
# ook een string zijn.

function groet(string $naam) : string
{
        $uur = date('H');

        if($uur >= 0 && $uur <= 6)
        {
                $begroeting = 'Goedenacht';
        }
        elseif($uur >= 6 && $uur <= 12)
        {
                $begroeting = 'Goedemorgen';
        }
        elseif($uur >= 12 && $uur <= 18)
        {
                $begroeting = 'Goedemiddag';
        }
        else
        {
                $begroeting = 'Goedeavond';
        }

        # De " . ' ' . "  is om twee
        # strings aan elkaar te plakken.
        #
        # Dit heet "concatinatie".
        return $begroeting . ' ' . $naam;
}

print groet("Stuart");
```

Een voordeel hiervan is dat je minder code hoeft te schrijven om te kijken of een functie wel teruggeeft wat je verwacht, en of de argumenten die je meegeeft ook zijn wat je ervan verwacht. Je kon, bij het PHP 5 voorbeeld hierboven, als argument voor $naam ook het getal 5 meegeven, dat werd ook geaccepteerd als geldig. Met deze variant gaat dat niet meer. Mocht je meer willen weten over de veranderingen in PHP 7 kun je de New features pagina bekijken op de website van PHP.

Objecten / OO programmeren

Met de introductie van PHP 5.0 in 2004 werd het in PHP ook mogelijk om objecten te gebruiken. Objecten kun je in zo'n beetje elke grote taal gebruiken (denk aan Java, C++, C# e.d) en zijn een manier om brokken programmacode te kunnen structureren en hergebruiken. PHP als taal gaat steeds meer gebruik maken van objecten en object-oriented programming (OOP). Om een voorbeeld te geven aan de hand van de `groet` functie hierboven:

```
<?php
# Print een begroeting naar het scherm
# afhankelijk van het tijdstip.
#
# Let op de manier waarop ik nu controleer
# hoe laat het is.

function groet($naam)
{
        $uur = new DateTime('now');
        $uur = $uur->format('H');

        if($uur >= 0 && $uur <= 6)
        {
                $begroeting = 'Goedenacht';
        }
        elseif($uur >= 6 && $uur <= 12)
        {
                $begroeting = 'Goedemorgen';
        }
        elseif($uur >= 12 && $uur <= 18)
        {
                $begroeting = 'Goedemiddag';
        }
        else
        {
                $begroeting = 'Goedeavond';
        }

        # De " . ' ' . "  is om twee
        # strings aan elkaar te plakken.
        #
        # Dit heet "concatinatie".
        return $begroeting . ' ' . $naam;
}
```

In `groet` is de variabele `$uur` nu een object, namelijk een `DateTime` object. Dit word een `instance` van een object genoemd.

Aangezien objecten en object georienteerd programmeren een hele andere tak van sport is laat ik het bij deze inleiding hierover. Wil je meer weten over OOP en objecten in PHP kun je die informatie vinden op de website van PHP in de sectie Clases and Objects.