Opleiding Applicatie Ontwikkelaar

Leerlijn Programmeren

Code indelen in PHP

Domein A Level 2

Auteur: Aminah Balfaqih

Datum: 11-12-2018

Inhoudsopgave

Overzicht

Level: Domein A Level 2
Duur: 32 uur
Methode: Weekplanning

Voorkennis

A1 Programmeren PHP basis. Als de Module A2 van Java al afgerond is zal je deze module aanzienlijk sneller doorlopen.

Materialen

- Je laptop met;
- Editor, bijvoorbeeld Kladblok, Notepad++, Atom (☺ : aanrader!)
- Webbrowser, bijvoorbeeld Internet Explorer, Firefox, Chrome
- XAMPP webserver (of andere servertool)

Instructies

- Het is aan te bevelen om deze module van begin tot eind te bestuderen. Sla geen stukken over, want dan mis je essentiële informatie.
- Probeer de voorbeeld code ook uit op je computer.

Beschrijving

Doelen

Na het bestuderen van deze module:

- ben je in staat om “nette” code te schrijven.
- Kan je gebruik maken van methoden/functies
- Kan je uitleggen waarom variabelen de ene keer wel bereikbaar zijn en de andere keer niet
- Kan je gebruik maken van recursieve methoden

Beoordeling

- Na het doorlopen van deze module heb je 3 inleveropdrachten gemaakt. Deze opdrachten dien je in te leveren via magister.
- De docent zal een mondeling houden over jou gemaakte werk.
- De opdrachten en de mondeling vormen samen jouw eindcijfer voor deze module.

Studieonderdelen

Gebruik van een methode

Tot nu toe heb je voornamelijk jouw code achtereenvolgend in een file geschreven. Om jouw code meer overzicht te geven is het van belang dat je methode/functies gaat gebruiken.

Hieronder kan je het verschil zien van een code die achter elkaar door geschreven wordt en hoe je dit kan omzetten naar methoden.

Voorbeeld zonder methode (voorbeeld 1):

```
<?php
  $dobbel =  rand(1,6);
  echo $dobbel;
?>
```

Voorbeeld met methode (voorbeeld 2):

```
<?php
  $dobbel = gooiDobbelsteen();
  echo $dobbel;

  function gooiDobbelsteen(){
   return rand(1,6);
  }
?>
```

Uitleg code:

In voorbeeld 1 is te zien dat de code elkaar opvolgt. Je kan je misschien wel voorstellen dat als je heel erg veel code hebt dat dit erg onoverzichtelijk kan worden. Daarom kan je een functie gebruiken om je code wat meer structuur te geven. Door methoden te gebruiken maak je code ook herbruikbaar.

In voorbeeld 2 kan je zien hoe de code opgedeeld met gebruik van een methode.

Dit doet de code:

- De *rand()* functie geeft een willekeurig getal tussen 1en 6.
- Met de opdracht return wordt dit getal teruggegeven als resultaat van de functie.

Oefeningen

Oefening 1

Maak een methode met de naam *maakVierkant* die als resultaat een string geeft. De string bevat een vierkant van 5 x 5 plussen. Als je de functie met een “echo” op het scherm zet moet je het volgende zien:

```
+++++
+++++
+++++
+++++
+++++
```

Oefening 2

Maak een methode met de naam evenDag die als resultaat geeft of de dag van vandaag een even getal is, resultaat is `true` of `false` (boolean).

Gebruik hiervoor de methode date() van php.

Oefening 3

Maak een methode met de naam getPHPosInfo die uit de systeemeigenschappen weergeeft van de machine waar php op draait.

Tip: http://php.net/manual/en/function.php-uname.php

Argumenten

Het voorbeeld van de dobbelsteen voert de code uit en geeft het resultaat terug. Hierbij wordt bij de aanroep van de methode geen informatie meegeven, maar dat wil je vaak wel. Dit kun je doen door argumenten te gebruiken.

Voorbeeld 1:

```
<?php
  $voorbeeld = 10;
  $resultaat = verhoogMetEen($voorbeeld);
  echo $voorbeeld;
  echo "<br/>";
  echo $resultaat;

  function verhoogMetEen($invoer){
   return ++$invoer;
  }
?>
```

Uitleg code:

De methode *verhoogMetEen* gebruikt de waarde die wordt meegegeven en verhoogt deze met 1, daarna wordt het resultaat weer teruggegeven.

- ++invoer verhoogt de waarde in invoer met 1 voor dat de waarde wordt gebruikt. Dit is wat anders dan invoer++, want daar wordt de waarde van invoer eerst gebruikt en daar pas verhoogt met 1. In dit voorbeeld zou invoer++ er voor zorgen dat de methode als resultaat de waarde teruggeeft dit in het argument *invoer* is meegegeven. Probeer dat zelf maar eens.
- ($invoer) geeft aan dat deze methode bij de aanroep gewacht dat er een waarde wordt meegegeven. Deze waarde wordt binnen de methode opgeslagen in de variabele $invoer. Deze variabele kun je alleen binnen deze methode gebruiken. Hierbuiten bestaat de variabele $invoer niet.

Voorbeeld 2:

```
<?php
  $a = 10;
  $b = 5;
  $som = telOp($a,$b);
  echo $som;

  function telOp($getal1, $getal2){
   return $getal1 + $getal2;
  }
?>
```

Uitleg code:

De methode *telOp* tel de twee argumenten bij elkaar op en geeft het resultaat (de som) terug.

- ($getal1, $getal2) de argumenten worden gescheiden door een komma.

Oefeningen

Oefening 4

Maak oefening 1 nogmaals, maar nu moet je de grootte van het vierkant als argument aan de methode kunnen meegeven.

Oefening 5

Maak een methode met de naam getMax die als argument een array van integer waarden krijgt en als resultaat de grootste waarde uit de lijst geeft.

Oefening 6

Maak een methode met de naam countChar die als argumenten een string en een char mee krijgt. De methode telt hoe vaak het teken in het tweede argument (char) in het eerste argument (String) voorkomt en geeft de waarde als resultaat.

Tip! *https://www.w3schools.com/php/php_string.asp*

Oefening 7

Maak een methode met de naam reverseString die als argument een string mee krijgt. Het resultaat van de methode is de invoer in omgekeerde volgorde, dus ‘oefening’ wordt ‘gninefeo’.

Tip! *https://www.w3schools.com/php/php_string.asp*

Include

In PHP is het mogelijk om je code over meerdere files te verdelen. Je kan dan jouw files aanroepen via de “include” methode. In een include file is het mogelijk om functies te groeperen die hetzelfde nut dienen. Als je bijvoorbeeld een rekenmachine maakt dan kan je alle functies die de sommen uitvoeren in een include file plaatsen. Dan staan ze allemaal netjes bij elkaar.

Als je een include file bovenin de PHP file zet dan kan je alle functies die in de include file staan gebruiken.

Voorbeeld:

Hoofd file

```
<?php
   include('includeTest.php');
   echo geefTekst();
?>
```

IncludeTest.php file

```
<?php
   function geefTekst(){
      return "Hello World!"
   }
?>
```

Uitleg van de code:

- In de hoofdfile wordt de include file aangeroepen door de regel: Include (includeTest.php);
- Zoals je kan zien staat er in de include file een functie “geefTekst()”.
- Omdat je de include functie gebruikt hebt kan je de functie in de include file gebruiken.

Include of require

Op het internet kom je zeker naast include ook functies tegen zoals bijvoorbeeld require. Hieronder de uitleg:

include	Include een file elke keer als de pagina geladen wordt.
include_once	Include de file alleen als hij nog niet ge-include is.
require	Include een file elke keer als de pagina geladen wordt.
require_once	Include de file alleen als hij nog niet ge-include is.

Zoals je ziet gedragen de include en de require zich hetzelfde. Er is echter wel een verschil. Als je een require gebruikt en er doet zich een fout voor dan zal het programma een fatal error geven. Het programma zal dan stoppen. Bij een include gaat de code echter door nadat er een waarschuwing gegeven is.

Recursie

Recursie is een truc om code van met name problemen met een wiskundige achtergrond simpeler te maken. Bij recursie wordt in een methode dezelfde methode weer aangeroepen, maar dan met een andere parameter. Dit klinkt ingewikkeld, dus bespreken we een voorbeeld.

Voorbeeld: vermenigvuldigen van twee getallen

Vermenigvuldigen is iets wat je al op de basisschool hebt geleerd. Nu gaan we er iets wiskundiger naar kijken. We stappen even bewust er over heen, dat vermenigvuldigen in een programmeertaal heel gemakkelijk is (er zijn trouwens nog steeds programmeertalen die niet kunnen vermenigvuldigen, dit heeft meestal met de simpelheid van het platform te maken)

Stel je voor: je wilt 4*3 berekenen, we weten allemaal dat de uitkomst 12 is, maar hoe bereken je dat?

Uitkomst = 4 + 4 + 4

Dus: je neemt 4 (1e keer) en dan tel je daar 4 bij op (2e keer) en daar bij tel je nog een keer 4 op. Je hebt dus 3 vieren bij elkaar opgeteld. Hieronder vind je de voorbeeld code:

```
function vermenigvuldig($aantalkeer, $getal) {
  //vermenigvuldigen is herhaalt optellen
  if ($aantalkeer == 0) {
    return 0;
  } else if ($aantalkeer == 1) {
    return $getal;
  } else {
    return vermenigvuldig($aantalkeer-1, $getal) + $getal;
  }
}
```

Uitleg

Stel je roept deze methode aan met `vermenigvuldig(0, 3)`:

1. aantalkeer is dan 0.
2. De methode keert dan terug met 0. Immers 0*3= 0

Daarna proberen we een aanroep met `vermenigvuldig(1, 3)`:

1. aantalkeer is 1
2. De methode keert dan terug met 3, zoals we geleerd hebben bij rekenen: 1*3= 3

Tot slot de laatste mogelijkheid, de aanroep met `vermenigvuldig(2, 3)`:

1. aantalkeer is 2
2. De methode roept zichzelf aan met aantalkeer – 1 (2-1=1)
3. In de tweede ronde geldt dus aantalkeer is 1
4. De tweede aanroep keert terug met antwoord 3
5. Daar tellen we 3 bij op (3 + 3 = 6)
6. De eerste aanroep keert terug met antwoord 6

Niet oneindig doorgaan

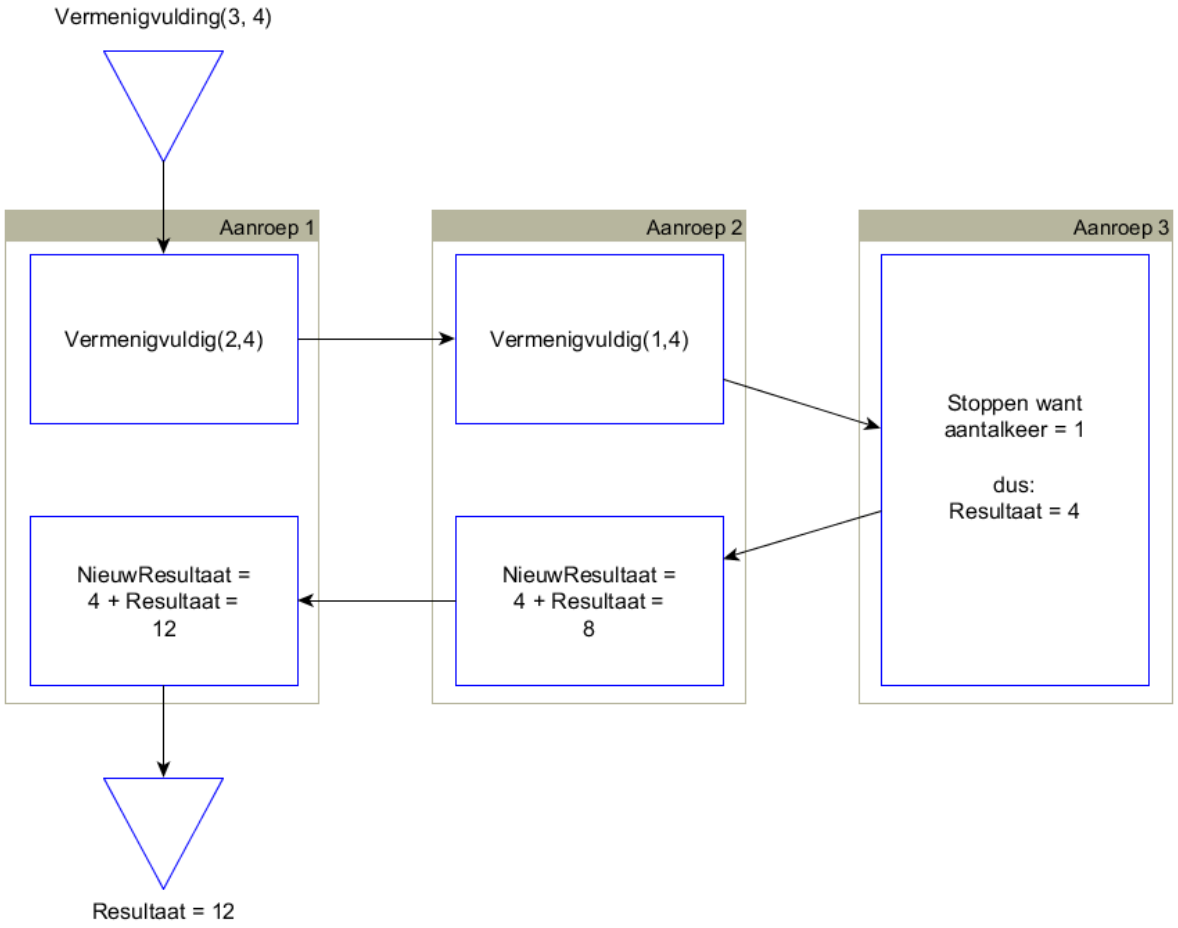
Een van de belangrijkste eigenschappen van een recursieve methode is dat er ook een stop-conditie in zit. Want anders zou het kunnen zijn dat je oneindig doorgaat, omdat je steeds weer dezelfde methode aanroept die zichzelf aanroept die zichzelf aanroept….

Uiteindelijk het programma of erger je computer vastloopt. In bovenstaande code wordt dit geregeld door de regels

```
if ($aantalkeer == 0) {
    return 0;
} else if ($aantalkeer == 1) {
    return $getal;
}
```

We weten dat er elke keer -1 gedaan wordt, en dus zul je een keer bij aantalkeer = 1 aankomen, en dan springt de methode weer terug naar buiten. En aantalkeer = 0 zorgt ervoor dat het goed gaat als je met 0 begint.

Schematisch gezien loopt het programma als volgt:



Oefening

Neem de code hieronder over en bedenk nog minimaal 3 vermenigvuldigen om te controleren of het klopt.

Gebruik de voorgaande functie vermenigvuldig() om onderstaande te testen.

```
<?php
  echo("4 * 3 = " . vermenigvuldig(4, 3));
  echo("6 * 5 = " . vermenigvuldig(6, 5));
?>
```

Voorbeeld 2: Faculteit berekenen

Met dezelfde manier kun je ook een faculteit berekenen (zie ook de bronnen)

- 0! = 1
- 1! = 1
- 2! = 1 *2
- 3! = 1 * 2 *3
- 4!= 1 * 2 * 3 *4
- Enzovoort

Oefening

Werk de faculteit berekening ook eens uit in PHP net zoals het voorbeeld hierboven.

Inleveropdrachten

Opdracht A

Maak een calculator die de volgende functies uit kan voeren:

- Optellen
- Aftrekken
- Vermenigvuldigen
- Delen
- Waarde omzetten naar negatief of positief (vermenigvuldigen met -1)

Zorg dat alle functies die je voor de calculatie gebruikt in een include file staan.

Opdracht B

Maak een include file waar je reguliere expressies kunt testen. De volgende onderdelen moet je kunnen testen:

- Is een variabele alleen alfabetisch
- Is een variabele alleen numeriek
- Nederlandse postcode check
- Nederland mobiel nummer check
- Is een variabele alleen “man” of “vrouw”
- Is een variabele een geldig email adres
- is een variabele een adres (alfabetisch opgevolgd door numeriek)

Schrijf elke onderdeel in een functie en geef deze functies een logische naam. Laat daarna via een hoofdfile zien dat de functies werken. Laat dus ook zien als de test faalt (foutmelding).

Hulp:

https://www.tutorialspoint.com/php/php_regular_expression.htm

http://php.net/manual/en/function.preg-match.php

http://www.rexegg.com/regex-php.html

Opdracht C: Fibonacci

Fibonacci was een van de eerste ontdekkers van een speciale reeks getallen:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, etc.

Deze kun je gemakkelijk berekenen: namelijk een getal is een optelling van 2 voorafgaande getallen.

1 + 1 = 2

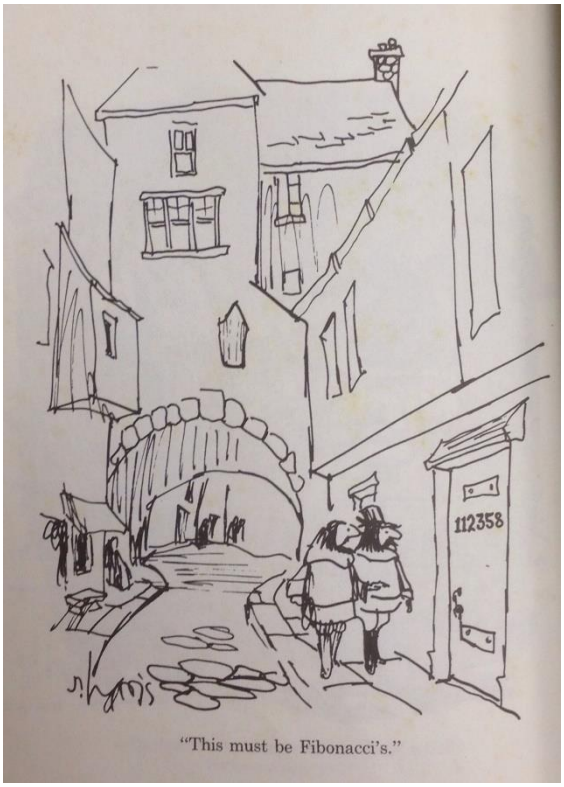
1 + 2 = 3

2 + 3 = 5

3 + 5 = 8

5 + 8 = 13 etc.

Deze reeks vindt onder andere een (wiskundige) toepassingen in de berekening over hoeveel konijnen je hebt na een aantal maanden. Dat gaat te ver in deze opdracht.



"This must be Fibonacci's."

Verbeter onderstaande code zodanig dat je een recursieve methode gebruikt om de getallen te berekenen. Als je goed kijkt, zie je ook dat er fouten in zitten. Die kun je verbeteren door de berekening goed te doen.

```
    //Fibonacci reeks is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

        echo("De Fibonacci reeks is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34");
        echo ("Element 0 = 0");
        echo ("Element 1 = 1");
        echo ("Element 2 = 1");
        echo ("Element 3 = 2");
        echo ("Element 4 = 3");
        echo ("Element 5 = 5");
        echo ("Element 6 = 9");
        echo ("Element 7 = 14");
        echo ("Element 8 = 23");
        echo ("Element 9 = 37");
```

Bronnen

- ✓ **Uitleg faculteit**
  http://www.megawetenschap.nl/faculteit.html

- ✓ **Meer uitleg over Fibonacci**
  https://nl.wikipedia.org/wiki/Rij_van_Fibonacci