

Snelstartgids PHP

Introductie in PHP?	2
Wat zou je al moeten weten?	2
Wat is PHP?	2
Wat is een PHP Bestand?	2
Wat is MySQL?	2
PHP + MySQL	2
Waarom PHP?	2
Waar te beginnen?	3
Installatie van PHP, Apache en MySQL database.	3
Wat heb je nodig?	3
Download PHP	3
Download MySQL Database	3
Download Apache Server	3
PHP Syntax	3
Commentaar in PHP	4
PHP variabelen	5
Variabelen in PHP	5
Regels Voor De Namen Van Variabelen	5
PHP operatoren	6
PHP If...Else Statements	8
Conditionele statements	8
De If...Else Statement	8
De ElseIf Statement	9
PHP Switch Statement	10
De Switch Statement	10
PHP Arrays	11
Wat is een array?	11
PHP Looping	14
PHP Functies	17
PHP Formulieren en Gebruikers invoer	20
PHP \$_GET	21
PHP \$_POST	22
PHP Date()	23
PHP Include File	25
PHP Bestands beheer	28
Openen van een Bestand	28
Een bestand sluiten	29
Controleer Einde Bestand (EOF)	29
Een Bestand Regel voor Regel Lezen	29
Een Bestand Character voor Character Lezen	29
PHP Cookies	31
Wat is een Cookie?	31
Hoe maak je een Cookie?	31
Hoe haal je een cookie op?	31
Hoe verwijder je een cookie?	32
Wat als een Browser Cookies Niet Ondersteund?	32
PHP Sessions	33
PHP Sessie Variabelen	33
Een PHP Sessie Starten	33
Opslaan van een Sessie Variabele	34
Vernietigen van een Sessie	34
PHP Verzenden van E-mail	35
De PHP mail() Functie	35
PHP Eenvoudige E-Mail	35
PHP Mail Formulier	36

Introductie in PHP?

Een PHP bestand mag tekst, HTML tags en scripts bevatten. Scripts in een PHP bestand worden vanaf de server uitgevoerd.

Wat zou je al moeten weten?

Voor je verder gaat zou je de basis van het onderstaande moeten begrijpen:

- HTML / XHTML
- Enige scripting kennis

Wat is PHP?

- PHP staat voor **PHP: Hypertext Preprocessor**
- PHP is een server-side scripting taal, zoals ASP
- PHP scripts worden gestart op de server
- PHP ondersteunt vele soorten databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is een open source software (OSS)
- PHP is gratis te downloaden en gebruiken

Wat is een PHP Bestand?

- PHP bestanden mogen tekst, HTML tags en scripts bevatten
- PHP bestanden worden als gewone HTML terug gestuurd naar de browser
- PHP bestanden kunnen de volgende bestands-extensie hebben ".php", ".php3", of ".phtml"

Wat is MySQL?

- MySQL is een database server
- MySQL is ideaal voor kleine en grote applicaties
- MySQL ondersteunt standaard SQL
- MySQL werkt op meerdere besturingssystemen
- MySQL is gratis te downloaden en gebruiken

PHP + MySQL

- PHP gecombineerd met MySQL zijn cross-platform (betekent dat je kan ontwikkelen in een Windows omgeving en uitvoeren op een Unix systeem)

Waarom PHP?

- PHP werkt op verschillende platformen (Windows, Linux, Unix, etc.)
- PHP is compatibel met bijna alle servers van vandaag (Apache, IIS, etc.)
- PHP is GRATIS te downloaden van de officiële PHP resource: www.php.net
- PHP is gemakkelijk te leren en werkt efficiënt op de server

Waar te beginnen?

- Installeer een Apache server op een Windows of Linux machine
- Installeer PHP op een Windows of Linux machine
- Installeer MySQL op een Windows of Linux machine

Installatie van PHP, Apache en MySQL database.

Wat heb je nodig?

Als jouw server PHP ondersteund – Hoef je niets te doen! Je hoeft niets te installeren – Maak gewoon een aantal .php bestanden in jouw webdirectory – en de server zal de pagina's automatisch parsen. De meeste web hosts ondersteunen PHP. Maar, als jouw server geen PHP ondersteund, moet je PHP installeren. Hieronder staat een link naar een goede uitleg van PHP.net over hoe je PHP5 kan installeren:
<http://www.php.net/manual/en/install.php>

Download PHP

Download PHP hier gratis: <http://www.php.net/downloads.php>

Download MySQL Database

Download MySQL hier gratis: <http://www.mysql.com/downloads/index.html>

Download Apache Server

Download Apache hier gratis: <http://httpd.apache.org/download.cgi>

PHP Syntax

Je kan PHP bron code niet bekijken door de knop "View source" in de browser te gebruiken – Je zal dan alleen de output van het PHP bestand zien, welke gewoon HTML is. Dit komt omdat de PHP scripts op de server worden gestart voordat het resultaat naar de browser wordt gestuurd.

Basis PHP Syntax

Een PHP script blok start altijd met `<?php` en eindigt met `?>`. Een PHP script blok kan overal in het document geplaatst worden.

Op een server waar shorthand ondersteuning aan staat voldoet: `<? en ?>`. Maar, om maximaal compatibel te zijn, raden we aan dat je de standaard vorm gebruikt. (`<?php`).

```
<?php  
?>
```

Een PHP bestand bestaat normaal uit HTML tags, zoals in een HTML bestand, en enige

PHP script code.

Hieronder, staat een voorbeeld van een simpel PHP script welke de tekst "Hallo Wereld" naar de browser stuurt:

```
<html>
  <body>
    <?php
      echo "Hallo Wereld";
    ?>
  </body>
</html>
```

Elke regel code in PHP moet eindigen met een puntkomma. De puntkomma is een scheiding die gebruikt wordt om de verschillende instructies te onderscheiden van elkaar.

Er zijn twee basis instructies om tekst weer te geven met PHP: **echo** en **print**. In het voorbeeld hierboven hebben we de echo instructie gebruikt om de tekst "Hallo Wereld" weer te geven.

Commentaar in PHP

In PHP, gebruiken we // om een enkele regel commentaar of /* en */ om een groter blok met commentaar weer te geven.

```
<html>
  <body>
    <?php
      //Dit is een regel commentaar
      /*
      Dit is
      een commentaar
      blok
      */
    ?>
  </body>
</html>
```

PHP variabelen

Variabelen worden gebruikt om waarden in op te slaan, zoals getallen, strings of functie resultaten, zodat deze meerdere keren in het script gebruikt kunnen worden.

Variabelen in PHP

Alle variabelen in PHP starten met een \$ symbool. Variabelen kunnen strings, getallen of arrays bevatten.

Hieronder, het PHP script wijst de string "Hallo Wereld" toe aan een variabele genaamd \$txt:

```
<html>
  <body>
    <?php
      $txt="Hallo Wereld";
      echo $txt;
    ?>
  </body>
</html>
```

Om twee of meer variabelen samen te voegen, gebruiken we de dot (.) operator:

```
<html>
  <body>
    <?php
      $txt1="Hallo Wereld";
      $txt2="1234";
      echo $txt1 . " " . $txt2 ;
    ?>
  </body>
</html>
```

Het resultaat van het script hierboven zal: "Hello World 1234" zijn.

Regels Voor De Namen Van Variabelen

- Een naam van een variabele moet starten met een letter of een underscore "_".
- Een naam van een variabele kan alleen alfanumerieke karakters en underscores (a-Z, 0-9, en _) bevatten.
- Een naam van een variabele mag geen spaties bevatten. Als een variabele uit meer dan één word moet bestaan dan moeten de woorden gescheiden worden met een underscore (\$my_string), of door hoofdletters te gebruiken(\$myString)

PHP operatoren

Operatoren worden gebruikt om met waarden te werken.

PHP Operatoren

Deze sectie geeft een overzicht in de verschillende operatoren die gebruikt kunnen worden in PHP.

Rekenkundige Operatoren

Operator	Omschrijving	Voorbeeld	Resultaat
+	Optellen	$x=2$ $x+2$	4
-	Aftrekken	$x=2$ $5-x$	3
*	Vermenigvuldigen	$x=4$ $x*5$	20
/	Delen	$15/5$ $5/2$	3 2.5
%	Modulus (Rest van delen)	$5\%2$ $10\%8$ $10\%2$	1 2 0
++	Vergroten	$x=5$ $x++$	$x=6$
--	Verkleinen	$x=5$ $x--$	$x=4$

Opdracht Operatoren

Operator	Voorbeeld	Hetzelfde als
=	$x=y$	$x=y$
+=	$x+=y$	$x=x+y$
-=	$x-=y$	$x=x-y$
=	$x=y$	$x=x*y$
/=	$x/=y$	$x=x/y$
%=	$x\%=y$	$x=x\%y$

Vergelijkings Operatoren

Operator	Omschrijving	Voorbeeld
==	is gelijk aan	$5==8$ geeft false
!=	is niet gelijk	$5!=8$ geeft true
>	is groter dan	$5>8$ geeft false
<	is kleiner dan	$5<8$ geeft true
>=	is groter dan of gelijk aan	$5>=8$ geeft false
<=	is kleiner dan of gelijk aan	$5<=8$ geeft true

Logische Operatoren

Operator	Omschrijving	Voorbeeld
&&	And(en)	x=6 y=3 (x < 10 && y > 1) geeft true
	Or(of)	x=6 y=3 (x==5 y==5) geeft false
!	Not(niet)	x=6 y=3 !(x==y) geeft true

PHP If...Else Statements

De **if**, **elseif** en **else** statements in PHP worden gebruikt om verschillende acties uit te voeren onder verschillende voorwaarden.

Conditionele statements

Vaak, als je code schrijft, wil je verschillende acties voor verschillende beslissingen laten uitvoeren.

Je kan conditionele statements in je code gebruiken om dit te doen.

- **if...else statement** – gebruik deze statement als je een stuk code wilt uitvoeren wanneer een conditie true(waar) is... en een ander stuk code als de conditie niet true(waar) is
- **elseif statement** – wordt gebruikt bij de if...else statement om een stukje code uit te voeren als één van meerdere condities true(waar) is.

De If...Else Statement

Als je een stuk code wilt uitvoeren al seen conditie true is en een ander stuk code als de conditie false is, gebruikt dan de if...else statement.

Syntax

```
if (conditie)
    code die gestart wordt als de conditie true is;
else
    code to die gestart wordt als de conditie false is;
```

Voorbeeld

Het volgende voorbeeld zal "Prettig weekend!" weergeven als het vandaag vrijdag is, anders zal het "Prettige dag!" weergeven.

```
<html>
<body>
  <?php
    $d=date("D");
    if ($d=="Fri")
      echo "Prettig weekend!";
    else
      echo "Prettige dag!";
  ?>
</body>
</html>
```

Als er meer dan één regel code uitgevoerd moet worden als een conditie true/false is, dan moet de uitvoer code tussen accolades gezet worden.

```
<body>
  <?php
    $d=date("D");
    if ($d=="Fri")
    {
      echo "Hallo!<br />";
      echo "Prettig weekend!";
      echo "Zie je maandag!";
    }
  ?>
</body>
```

De ElseIf Statement

Als je een stuk codewil uitvoeren als één van meerder condities true(waar) is gebruik je de elseif statement.

Syntax

```
if (conditie)
    Code die wordt uitgevoerd als de conditie true is;
elseif (conditie)
    Code die wordt uitgevoerd als de conditie true is;
else
    Code die wordt uitgevoerd als de conditie false is;
```

Voorbeeld

Het volgende voorbeeld zal "Prettig weekend!" weergeven als het vandaag vrijdag is en "Prettige Zondag!" Als het vandaag zondag is. Anders zal er "Prettige dag!" weergegeven worden:

```
<html>
<body>
  <?php
    $d=date("D");
    if ($d=="Fri")
        echo "Prettig weekend!";
    elseif ($d=="Sun")
        echo "Prettige zondag!";
    else
        echo "Prettige dag!";
  ?>
</body>
</html>
```

PHP Switch Statement

De Switch statement in PHP wordt gebruikt om één van meerdere acties uit te voeren gebaseerd op een of meerdere condities.

De Switch Statement

Als je één van vele blokjes code wil uitvoeren gebruik je de switch statement. De switch statement wordt gebruikt om lange blokken van if..elseif..else code te voorkomen.

Syntax

```
switch (expressie)
{
case label1:
    code wordt uitgevoerd als expressie = label1;
    break;
case label2:
    code wordt uitgevoerd als expressie = label2;
    break;
default:
    code wordt uitgevoerd
    als expressie anders is
    dan beide labels, label1 en label2;
}
```

Voorbeeld

Zo werkt het:

- Een enkele expressie (vaak een variabele) wordt eenmalig geëvalueerd.
- De waarde van de expressie wordt vergeleken met de waarden van elke case.
- Als de vergelijking waar is zal de bijbehorende code worden uitgevoerd.
- Nadat de code wordt uitgevoerd, wordt, **break** gebruikt om te voorkomen dat de code van de volgende case wordt gebruikt.
- Het default statement is wordt gebruikt als geen van de cases waar is.

```
<html>
<body>
  <?php
    switch ($x)
    {
      case 1:
        echo "Nummer 1";
        break;
      case 2:
        echo "Nummer 2";
        break;
      case 3:
        echo "Nummer 3";
        break;
      default:
        echo "Geen nummer tussen 1 en 3";
    }
  ?>
</body>
</html>
```

PHP Arrays

Een array kan meerdere waarden opslaan in een enkele variabele naam.

Wat is een array?

Wanneer je met PHP werkt, gebeurt het vroeg of laat dat je vele zelfde variabelen wilt aanmaken.

In plaats hiervan kan je de data in de vorm van elementen opslaan in een array.

Elk element in de array heft zijn eigen ID zodat deze makkelijk bereikbaar is.

Er zijn drie verschillende soorten arrays:

- **Numerieke array** - Een array met een numerieke ID sleutel
- **Associative array** - Een array waar elke ID sleutel wordt geassocieerd met een waarde.
- **Multidimensionale array** - Een array die meerdere arrays bevat.

Numerieke Arrays

Een numerieke array slaat elk element op met een numerieke ID sleutel(key).

Er zijn verschillende manieren om een numerieke array te maken.

Voorbeeld 1

In dit voorbeeld wordt de ID key automatisch toegewezen:

```
$names = array("Peter","Quagmire","Joe");
```

Voorbeeld 2

In dit voorbeeld wijzen we handmatig een waarde toe aan de ID key:

```
$names[0] = "Peter";  
$names[1] = "Quagmire";  
$names[2] = "Joe";
```

De ID keys kunnen gebruikt worden in een script script:

```
<?php  
$names[0] = "Peter";  
$names[1] = "Quagmire";  
$names[2] = "Joe";  
echo $names[1] . " en " . $names[2] .  
" zijn " . $names[0] . "'s burens";  
?>
```

De uitvoer van bovenstaande code:

```
Quagmire en Joe zijn Peter's burens
```

Associative Arrays

Een associatieve array, elke ID key wordt geassocieerd met een waarde.

Met het opslaan van data met specifieke namen is een numerieke array niet altijd de beste manier om dat te doen.

Met associatieve arrays kunnen we waarden als key gebruiken en daar weer waarden aan toe wijzen.

Voorbeeld 1

In dit voorbeeld gebruiken we een array om leeftijden te koppelen aan verschillende personen:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

Voorbeeld 2

Dit voorbeeld is hetzelfde als voorbeeld1, maar laat een andere manier zien om de array te maken:

```
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";
```

De ID keys kunnen gebruikt worden in een script:

```
<?php  
$ages['Peter'] = "32";  
$ages['Quagmire'] = "30";  
$ages['Joe'] = "34";  
echo "Peter is " . $ages['Peter'] . " jaar oud."  
?>
```

Bovenstaande code zal het volgende weergeven:

```
Peter is 32 jaar oud.
```

Multidimensionale Arrays

In een multidimensionale array, kan elke element in de hoofd array ook een array zijn. En elk element in een sub-array kan ook weer een array zijn, en zo verder.

Voorbeeld

In dit voorbeeld maken we een multidimensionale array, met automatisch toegewezen ID keys:

```
$families = array  
(  
  "Griffin"=>array  
  (  
    "Peter",  
    "Lois",  
    "Megan"  
  ),  
  "Quagmire"=>array  
  (  
    "Glenn"  
  ),  
  "Brown"=>array  
  (  
    "Cleveland",  
    "Loretta",  
    "Junior"  
  )  
);
```

Bovenstaande array kan je ook zo bekijken:

```
Array
(
  [Griffin] => Array
  (
    [0] => Peter
    [1] => Lois
    [2] => Megan
  )
  [Quagmire] => Array
  (
    [0] => Glenn
  )
  [Brown] => Array
  (
    [0] => Cleveland
    [1] => Loretta
    [2] => Junior
  )
)
```

PHP Looping

Looping statements in PHP worden gebruikt om een zelfde blok code een zeker aantal keer uit te voeren.

Looping

Vaak wil je, als je code schrijft, een blok code een aantal keer laten uitvoeren. Je kan looping statements in je code gebruiken om dit te doen.

In PHP hebben we de volgende looping statements:

- **while** – herhaalt de code zolang een conditie true(waar) is.
- **do...while** – loopt een keer door de code en herhaalt de code zolang de conditie true(waar) is.
- **for** – herhaalt de code een bepaald aantal keer.
- **foreach** – herhaalt de code voor elk element in een array.

De while Statement

De while statement zal een blok code uitvoeren, **als en zo lang als** een conditie true(waar) is.

Syntax

```
while (condition)
    Code die uitgevoerd moet worden;
```

Voorbeeld

Hetvolgende voorbeeld demonstreerd een loop die doorgaat zolang de variabele i is kleiner dan, of gelijk aan 5 is, i wordt bij elke loop 1 groter:

```
<html>
<body>
<?php
    $i=1;
    while($i<=5)
    {
        echo "Het number is " . $i . "<br />";
        $i++;
    }
?>
</body>
</html>
```

De do...while Statement

De do...while statement zal de code op zijn minst één keer uitvoeren – de code wordt herhaalt zolang de conditie true (waar) is.

Syntax

```
do
{
    code die uitgevoerd wordt;
}
while (condition);
```

Voorbeeld

Het volgende voorbeeld zal de waarde van `i` in ieder geval één keer met 1 verhogen, en dat blijven doen zolang de waarde van `i` lager als 5 is:

```
<html>
<body>
  <?php
    $i=0;
    do
    {
      $i++;
      echo "Het number is " . $i . "<br />";
    }
    while ($i<5);
  ?>
</body>
</html>
```

De for Statement

De for statement wordt gebruikt als je weet hoeveel keer je een blok code wilt uitvoeren.

Syntax

```
for (initialization; condition; increment)
{
  Code die uitgevoerd moet worden;
}
```

N.b.: De for statement heft drie parameters. De eerste parameter initialiseert de variablen, de tweede parameter bevat de conditie, en de derde parameter bevat de waarde waarmee de variablen wordt veranderd. Als er meerdere variabelen worden gebruikt moeten de increment parameters gescheiden worden met komma's. De conditie moet uitkomen op true of false (waar of nietwaar).

Voorbeeld

Het volgende voorbeeld geeft de tekst "Hallo Wereld!" vijf keer weer:

```
<body>
  <?php
    for ($i=1; $i<=5; $i++)
    {
      echo "Hallo Wereld!<br />";
    }
  ?>
</body>
```

De foreach Statement

De foreach statement wordt gebruikt om door arrays te lopen.

Bij elke loop, zal de waarde van het huidige array element in `$value` worden gezet (en de array pointer wordt één verplaatst) en zo verder de volgende loop, met het volgende element.

Syntax

```
foreach (array as value)
{
  Code die moet worden uitgevoerd;
}
```

Voorbeeld

Het volgende voorbeeld laat een loop zien de de waarden in de array zal weergeven:

```
<html>
<body>
  <?php
    $arr=array("one", "two", "three");
    foreach ($arr as $value)
    {
      echo "Value: " . $value . "<br />";
    }
  ?>
</body>
</html>
```

PHP Functies

De echte kracht van PHP zit in de functies.

In PHP – zijn meer dan 700 ingebouwde functies voorhanden.

Een PHP Functie maken

Een functie is een blok code welke gestart kan worden zodra het nodig is.

- Alle functies beginnen met het woord "function()"
- Geef de functie een naam – Je zou aan de naam van de functie moeten kunnen zien wat de functie doet. De naam kan beginnen met een letter of een underscore (niet met een nummer)
- Voeg een "{" toe - De functie code begint na de opening accolade
- Voeg de functie code toe
- Voeg een "}" toe - De functie eindigt met een sluit accolade

Voorbeeld

Een eenvoudige functie die mijn naam schrijft wanneer hij wordt aangeroepen:

```
<body>
<?php
function writeMyName()
{
    echo "Erik_P";
}
writeMyName();
?>
</body>
```

Een PHP Functie gebruiken

Nu zullen we de functie gebruiken in een PHP script:

```
<body>
<?php
function writeMyName()
{
    echo "Erik_P";
}
echo "Hallo Wereld!<br />";
echo "Ik ben ";
writeMyName();
echo ".<br />Ja, ja, ";
writeMyName();
echo " dat ben ik.";
?>
</body>
```

De uitvoer van de bovenstaande code:

```
Hallo Wereld!
Ik ben Erik_P.
Ja, ja, Erik_P dat ben ik.
```

PHP Functie – Parameters toevoegen

Onze eerste functie (writeMyName()) is een zeer eenvoudige functie. Het schrijft alleen een statische string.

Om meer functionaliteit toe te voegen aan de functie kunnen we parameters toevoegen.

Een parameter is net zoals een variabele.

De parameters kunnen tussen de haken achter de functie naam gespecificeerd worden.

Voorbeeld 1

Het volgende voorbeeld zal verschillende voornamen schrijven, maar dezelfde achternaam:

```
<html>
<body>
<?php
function writeMyName($fname)
{
echo $fname . "_P.<br />";
}
echo "Ik ben ";
writeMyName("Erik");
echo "Ik ben ";
writeMyName("Sjaak");
echo "Ik ben ";
writeMyName("Joep");
?>
</body>
</html>
```

De uitvoer van de bovenstaande code:

```
Ik ben Erik_P.
Ik ben is Sjaak_P.
Ik ben is Joep_P.
```

Voorbeeld 2

De volgende functie heeft twee parameters:

```
<html>
<body>
<?php
function writeMyName($fname,$punctuation)
{
echo $fname . "_P" . $punctuation . "<br />";
}
echo "Ik ben ";
writeMyName("Erik",".");
echo "Ik ben ";
writeMyName("Sjaak","!");
echo "Ik ben ";
writeMyName("Joep","...");
?>
</body>
```

De uitvoer van de bovenstaande code:

```
Ik ben Erik_P.
Ik ben Sjaak_P!
Ik ben Joep_P...
```

PHP Functies - Return waarden

Functies kunnen ook gebruikt worden om waarden terug te geven aan het script.

Voorbeeld

```
<html>
<body>
<?php
function add($x,$y)
{
$total = $x + $y;
return $total;
}
echo "1 + 16 = " . add(1,16)
?>
</body>
</html>
```

De uitvoer van de bovenstaande code:

```
1 + 16 = 17
```

PHP Formulieren en Gebruikers invoer

De PHP `$_GET` en `$_POST` variabelen worden gebruikt om informatie te ontvangen van formulieren, zoals gebruikers invoer.

PHP Formulier Verwerking

Het belangrijkste wat opvalt met HTML formulieren en PHP is dat elk formulier element automatisch beschikbaar is in jouw PHP scripts.

Formulier voorbeeld:

```
<html>
<body>
  <form action="welkom.php" method="post">
    Naam: <input type="text" name="name" />
    Leeftijd: <input type="text" name="age" />
    <input type="submit" />
  </form>
</body>
</html>
```

De voorbeeld HTML pagina hierboven bevat twee tekst velden en een submit knop. Als een gebruiker dit formulier invuld en op de submit knop klikt, zal de formulier data naar het "welkom.php" bestand gestuurd worden.

Het "welkom.php" bestand ziet er zo uit:

```
<html>
<body>
  Welkom <?php echo $_POST["name"]; ?>.<br />
  Je bent<?php echo $_POST["age"]; ?> jaar oud.
</body>
</html>
```

Een voorbeeld van de scherm weergave:

```
Welkom John.
Je bent 28 jaar oud.
```

De PHP `$_GET` en `$_POST` variabelen worden in de volgende hoofdstukken uit gelegd.

Formulier Validatie

Gebruikers invoer moet gevalideerd worden in de browser, waar mogelijk (door client scripts (JavaScript)). Browser validatie is sneller en je verlaagd de server belasting. Je moet overwegen om via de server te valideren als de gegevens een database in moeten.

PHP \$_GET

De \$_GET variabele wordt gebruikt op formulier waarden op te halen met de "get" methode.

De \$_GET Variabele

De \$_GET variabele is een array van variabele namen en waarden verzonden met de HTTP GET methode.

De \$_GET variabele wordt gebruikt om de waarden op te halen van een formulier welke verzonden wordt met het attribuut method="get". Informatie die wordt verzonden met de GET methode is voor iedereen zichtbaar (Het wordt weergegeven in de adres balk van de browser) en wordt gelimiteerd in de hoeveelheid informatie die verzonden kan worden (max. 100 karakters).

Voorbeeld

```
<form action="welkom.php" method="get">
  Naam: <input type="text" name="name" />
  Leeftijd: <input type="text" name="age" />
  <input type="submit" />
</form>
```

Wanneer de gebruiker op de "Submit" button klikt, zal de verzonden URL er ongeveer zo uit zien:

```
http://www.domein.nl/welkom.php?name=Erik_P&age=33
```

Het "welkom.php" bestand kan nu de \$_GET variabele gebruiken om de verzonden data op te halen (Merk op dat de formulier veld namen automatisch de ID sleutels zijn in de \$_GET array):

```
Welkom <?php echo $_GET["name"]; ?>.<br />
Je bent <?php echo $_GET["age"]; ?> jaar oud!
```

Waarom de \$_GET methode gebruiken?

N.b.: Als de \$_GET variabele wordt gebruikt dan zijn alle variabele namen en waarden zichtbaar in de URL. Dus deze methode moet niet gebruikt worden om wachtwoorden en andere gevoelige informatie te verzenden! Maar, omdat de variabelen in de URL staan is het mogelijk om de pagina op te slaat in de favorieten(Bookmarks), dit kan af en toe handig zijn.

N.b.: De HTTP GET methode is niet geschikt voor grote variabele waarden; De waarde kan niet groter zijn dan 100 karakters.

De \$_REQUEST Variabele

De PHP \$_REQUEST variabele bevat de inhoud van \$_GET, \$_POST en \$_COOKIE. De PHP \$_REQUEST variabele kan gebruikt worden om het resultaat op te halen van formulier data verzonden met de GET en de POST methode.

Voorbeeld

```
Welkom <?php echo $_REQUEST["name"]; ?>.<br />
Je bent <?php echo $_REQUEST["age"]; ?> jaar oud!
```

PHP \$_POST

De `$_POST` variabele wordt gebruikt om waarden op te halen van een formulier met de `method="post"` methode.

De `$_POST` Variabele

De `$_POST` variabele is een array van variabele namen en waarden verzonden met de HTTP POST methode.

De `$_POST` variabele wordt gebruikt om formulier waarden op te halen verzonden met de `method="post"` methode. Informatie verzonden met de POST methode is onzichtbaar voor anderen en heft geen limiet met de hoeveelheid data die verzonden wordt.

Voorbeeld

```
<form action="welcome.php" method="post">
  Vul je naam in: <input type="text" name="name" />
  Vul je leeftijd in: <input type="text" name="age" />
  <input type="submit" />
</form>
```

Wanneer de gebruiker op de "Submit" button klikt, zal de URL geen formulier data bevatten en zal er ongeveer zo uit zien:

```
http://www.domein.nl/welkom.php
```

Het "welkom.php" bestand kan nu de `$_POST` variable gebruiken om de data op te halen (merk op dat de formulier veldnamen automatisch de ID sleutels in de `$_POST` array worden.):

```
Welkom <?php echo $_POST["name"]; ?>.<br />
Je bent <?php echo $_POST["age"]; ?> jaar oud!
```

Waarom `$_POST` gebruiken?

- Variabelen verzonden met HTTP POST zijn niet zichtbaar in de URL
- De variabelen hebben geen limiet voor de lengte

Maar, omdat de variabelen niet zichtbaar zijn in de URL, is het niet mogelijk om de pagina op te slaan in de favorieten(Bookmark).

De `$_REQUEST` Variabele

De PHP `$_REQUEST` variabele bevat de inhoud van `$_GET`, `$_POST` en `$_COOKIE`. De PHP `$_REQUEST` variabele kan gebruikt worden om het resultaat op te halen van formulier data verzonden met de GET en de POST methode.

Voorbeeld

```
Welkom <?php echo $_REQUEST["name"]; ?>.<br />
Je bent <?php echo $_REQUEST["age"]; ?> jaar oud!
```

PHP Date()

De PHP date() functie wordt gebruikt om met tijd en datum te werken.

De PHP Date() Functie

De PHP date() functie maakt van een timestamp een beter leesbare tijd en datum.

Syntax

```
date(format, timestamp)
```

Parameter	Omschrijving
format	Verplicht. Specificeert het formaat van de timestamp
timestamp	Optioneel. Specificeert een timestamp. Standaard de huidige tijd en datum.(als timestamp)

PHP Date - Wat is een Timestamp?

Een timestamp is het aantal seconden sinds 1 Januari 1970 om 00:00:00 GMT. Dit is ook bekend als de Unix Timestamp.

PHP Date – Datum weergave

De eerste parameter in de date() functie specificeert hoe de datum/tijd moet worden weergegeven. Het gebruikt letters om tijd en datum formaten weer te geven. Hier zijn enkele letters die gebruikt kunnen worden:

- d – De dag van de maand (01-31)
- m – De huidige maand, als getal (01-12)
- Y – Het huidige jaar in vier getallen

Andere karakters, zoals "/", ".", of "-" kunnen ook gebruikt worden tussen de letters om de delen van de tijd/datum te scheiden:

```
<?php
echo date("d/m/Y");
echo "<br />";
echo date("d.m.Y");
echo "<br />";
echo date("d-m-Y");
?>
```

De uitvoer van de bovenstaande code:

```
18/01/2007
18.01.2007
18-01-2007
```

PHP Date – Een Timestamp Toevoegen

De tweede parameter in de date() functie specificeert een timestamp. Dit is een optionele parameter. Als je geen timestamp toevoegd zal de huidige tijd gebruikt worden.

In ons volgende voorbeeld gebruiken we de mktime() functie om een timestamp voor morgen te maken. De mktime() functie geeft de Unix timestamp voor een gegeven datum terug.

Syntax

```
mktime(hour, minute, second, month, day, year, is_dst)
```

Om een dag in de toekomst te gaan kan je simpelweg een dag optellen bij het day argument van mktime():

```
<?php
$morgen = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Morgen is ".date("Y/m/d/",$morgen);
?>
```

De uitvoer zal er dan zo uitzien:

```
Morgen is het 19/01/2007
```

PHP Include File

Server Side Includes (SSI) worden gemaakt om functies, headers, footers of elementen te hergebruiken op meerdere paginas.

Server Side Includes

Je kan de inhoud van een bestand aan een PHP bestand toevoegen voordat deze op de server wordt gestart, met de include() of de require() functie. De twee functies zijn hetzelfde, behalve hoe ze omgaan met fouten. De include() functie geeft een waarschuwing (maar het script zal wel doorgaan met de uitvoer), terwijl de require() functie een fatal error geeft (het script stopt na de fout).

Deze twee functies worden gebruikt om functies, headers, footers of elementen te maken die gebruikt kunnen worden op meerdere webpagina's.

Dit kan de ontwikkelaar een behoorlijke tijds winst opleveren. Dit betekent dat je een standaard header of menu bestand kan maken welke je in al je webpagina's kan toevoegen. Wanneer je dan bijvoorbeeld de header van je site wilt veranderen hoef je maar één bestand te veranderen (en niet op elke pagina van je site).

De include() Functie

De include() functie neemt alle tekst in een bepaald bestand en kopiëert dat in het bestand dat de include functie gebruikt.

Voorbeeld 1

Stel je voor dat je een standard header bestand hebt, genaamd "header.php". Om dit header bestand in een pagina op te nemen, kan je de include() functie, zo gebruiken:

```
<html>
<body>
<?php include("header.php"); ?>
<h1>Welkom op mijn homepage</h1>
<p>Hier komt de tekst van je pagina</p>
</body>
</html>
```

Voorbeeld 2

Stel je nu voor dat je een standard menu bestand hebt welke op al je pagina's moet worden weergegeven (include bestanden hebben normaal een ".php" extensie). Bekijk het "menu.php" bestand hieronder:

```
<html>
<body>
<a href="http://www.domein.nl/default.php">Home</a> |
<a href="http://www.domein.nl/over.php">Over ons</a> |
<a href="http://www.domein.nl/contact.php">Contact</a>
```

De drie bestanden, "default.php", "over.php", en "contact.php" moeten allen het menu.php" bestand includen. Hier is de code van "default.php":

```
<?php include("menu.php"); ?>
<h1>Welkom op mijn homepage</h1>
<p>Hier komt de tekst van je pagina</p>
</body>
</html>
```

Als je de broncode van "default.php" bekijkt, zal je ongeveer dit zien:

```
<html>
<body>
  <a href="default.php">Home</a> |
  <a href="over.php">Over ons</a> |
  <a href="contact.php">Contact</a>
  <h1>Welkom op mijn homepage</h1>
  <p>Hier komt de tekst van je pagina</p>
</body>
</html>
```

En natuurlijk moeten we hetzelfde doen met de "about.php" en "contact.php" bestanden. Door include bestanden te gebruiken, kan je simpelweg de tekst in het "menu.php" bestand veranderen, als je een link wilt veranderen of.

De require() Functie

De require() functie is identiek aan include(), alleen de fout afhandeling werkt anders. De include() functie geeft een waarschuwing (maar het script gaat door met de uitvoer), terwijl de require() functie een fatale fout (fatal error) geeft (het script stopt met de uitvoer).

Als je een fout krijgt met de include() functie, kan je een foutmelding krijgen zoals hieronder:

PHP code:

```
<html>
<body>
  <?php
    include("verkeerde_bestand.php");
    echo "Hallo Wereld!";
  ?>
</body>
</html>
```

Fout melding:

```
Warning: include(verkeerde_bestand.php) [function.include]:
failed to open stream:
No such file or directory in C:\home\website\test.php on line 5
Warning: include() [function.include]:
Failed opening 'verkeerde_bestand.php' for inclusion
(include_path='.;C:\php5\pear')
in C:\home\website\test.php on line 5
Hallo Wereld!
```

Merk op dat de echo statement nog wel wordt uitgevoerd!

Nu, hetzelfde voorbeeld met de require() functie.

PHP code:

```
<html>
<body>
  <?php
    require("verkeerde_bestand.php");
    echo "Hallo Wereld!";
  ?>

</body>
</html>
```

Fout melding:

```
Warning: require(verkeerde_bestand.php) [function.require]:  
failed to open stream:  
No such file or directory in C:\home\website\test.php on line 5  
Fatal error: require() [function.require]:  
Failed opening required 'verkeerde_bestand.php'  
(include_path='.;C:\php5\pear')  
in C:\home\website\test.php on line 5
```

De echo statement werd niet uitgevoerd omdat de uitvoer van het script stopt na een fatale fout.

Het wordt aangeraden altijd de require() functie in plaats van de include() functie te gebruiken, omdat we eigenlijk niet willen dat een script doorgaat als er bestanden missen.

PHP Bestands beheer

De `fopen()` functie wordt gebruikt om bestanden te openen in PHP.

Openen van een Bestand

De `fopen()` functie wordt gebruikt om bestanden te openen in PHP.

De eerste parameter van deze functie bevat de naam van het bestand dat geopend moet worden. De tweede parameter specificeert in welke modus het bestand geopend moet worden:

```
<html>
<body>
<?php
$file=fopen("Welkom.txt","r");
?>
</body>
</html>
```

Het bestand kan in één van onderstaande modi worden geopend:

Modes	Description
r	Alleen lezen. Start aan het begin van het bestand.
r+	Lezen/Schrijven. Start aan het begin van het bestand
w	Alleen schrijven. Opent en wist de inhoud van het bestand; of maakt nieuw bestand als het niet bestaat.
w+	Lezen/Schrijven. Opent en wist de inhoud van het bestand; of maakt nieuw bestand als het niet bestaat.
a	Toevoegen. Opent en schrijft aan het einde van het bestand, of maakt nieuw bestand als het niet bestaat.
a+	Lezen/Toevoegen. Bewaard inhoud door aan het einde van het bestand te schrijven.
x	Alleen Schrijven. Maakt een nieuw bestand. Geeft FALSE terug en een error als het bestand al bestaat.
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

N.b.: Als de `fopen()` functie een bepaald bestand niet kan openen, zal er 0 (false) worden terug gegeven.

Voorbeeld

Het volgende voorbeeld genereert een bericht als de `fopen()` functie het bestand niet kan openen:

```
<html>
<body>
<?php
$file=fopen("welkom.txt","r") or exit("Kan bestand niet openen!");
?>
</body>
</html>
```

Een bestand sluiten

De `fclose()` functie wordt gebruikt om een geopend bestand te sluiten:

```
<?php
$file = fopen("test.txt","r");
//enige code die uitgevoerd moet worden
fclose($file);
?>
```

Controleer Einde Bestand (EOF)

De `feof()` functie controleert of de "end-of-file" (EOF) is bereikt.

De `feof()` functie is handig om door data te lopen waarvan je niet weet hoelang het is.

N.b.: Je kan niet van bestanden lezen geopend in in `w`, `a`, and `x` mode!

```
if (feof($file)) echo "End of file";
```

Een Bestand Regel voor Regel Lezen

De `fgets()` functie wordt gebruikt om een enkele regel van een bestand te lezen.

N.b.: Na een aanroep met deze functie wordt de bestands pointer een regel verder gezet.

Voorbeeld

Het onderstaande voorbeeld leest een bestand regel voor regel, totdat het einde van het bestand wordt bereikt:

```
<?php
$file = fopen("welkom.txt", "r") or exit("Kan bestand niet openen!");
//geef regel voor regel weer tot einde bestand
while(!feof($file))
{
    echo fgets($file). "<br />";
}
fclose($file);
?>
```

Een Bestand Character voor Character Lezen

De `fgetc()` functie wordt gebruikt om een enkele character te lezen van een bestand.

N.b.: Na een aanroep met deze functie wordt de bestands pointer een character verder gezet.

Voorbeeld

Het onderstaande voorbeeld leest een bestand character voor character tot het einde van het bestand is bereikt:

```
<?php
$file=fopen("welkom.txt","r") or exit("Kan bestand niet openen!");
while (!feof($file))
{
    echo fgetc($file);
}
fclose($file);
?>
```

PHP Cookies

Een cookie wordt vaak gebruikt om een gebruiker te indentificeren.

Wat is een Cookie?

Een cookie wordt vaak gebruikt om een gebruiker te indentificeren. Een cookie is een klein bestandje welke de server op de computer van de gebruiker zet. Elke keer dat de zelfde computer een pagina opvraagt met de browser, zal deze de cookie ook verzenden. Met PHP kan je cookies maken en opvragen.

Hoe maak je een Cookie?

De setcookie() functie wordt gebruikt om een cookie te maken.

N.b.: De setcookie() functie moet VOOR de <html> tag gebruikt worden.

Syntax

```
setcookie(name, value, expire, path, domain);
```

Voorbeeld

In het onderstaande voorbeeld, maken we een cookie genaamd "user" en wijzen we de waarde "Erik_P" toe aan de cookie. Ook stellen we in dat de cookie na eer uur verloopt:

```
<?php
setcookie("user", "Erik_P", time()+3600);
?>
<html>
<body>
</body>
</html>
```

N.b.: De waarde van de cookie wordt automatisch URLgecodeerd wanneer de cookie verzonden wordt, en automatisch gedecodeerd wanneer de cookie opgevraagd wordt. (om URLcoderen te voorkomen gebruik je: setrawcookie()).

Hoe haal je een cookie op?

De PHP \$_COOKIE variabele wordt gebruikt om een cookie waarde op te halen.

In het onderstaande voorbeeld, halen we de waarde op van de cookie genaamd "user" op, en geven het weer op de pagina:

```
<?php
// Print een cookie
echo $_COOKIE["user"];
// Mannier om alle cookies te bekijken
print_r($_COOKIE);
?>
```

In het volgende voorbeeld gebruikten we de `isset()` functie om uit te vinden of er een cookie bestaat:

```
<html>
<body>
<?php
if (isset($_COOKIE["user"]))
    echo "Welkom" . $_COOKIE["user"] . "!\n";
else
    echo "Welkom Gast!\n";
?>
</body>
</html>
```

Hoe verwijder je een cookie?

Als je een cookie wilt verwijderen zet je de verloop datum in het verleden.

Voorbeeld voor verwijderen:

```
<?php
// stel de verloop datum op een uur geleden
setcookie("user", "", time()-3600);
?>
```

Wat als een Browser Cookies Niet Ondersteund?

Als jouw programma met een browser te maken krijgt die cookies niet ondersteund, moet je een andere methode gebruiken om informatie van de ene pagina naar de andere pagina te leiden. Een methode is het gebruik van formulieren (is in een eerder artikel besproken).

PHP Sessions

Een PHP sessie variabele wordt gebruikt om informatie op te slaan, of instellingen te wijzigen voor de huidige sessie van de gebruiker. Sessie variabelen houden informatie over een enkele gebruiker vast, en zijn in alle pagina's binnen de applicatie beschikbaar.

PHP Sessie Variabelen

Als je met een applicatie bezig bent, open je deze, maak je wat wijzigingen, en sluit je het. Dit lijkt op een sessie. De computer weet wie je bent, wanneer je de applicatie bent gestart, en wanneer je bent gestopt. Maar op het internet hebben we een probleem: De web server weet niet wie je bent en wat je doet, omdat het HTTP adres dit niet onthoudt.

Een PHP sessie lost dit probleem op, door informatie op de server op te slaan voor later gebruik. (b.v. gebruikers naam, webshop items, etc). Sessie informatie is tijdelijk, als de gebruiker de site verlaat gaat de data verloren.

Sessies werken door een unieke ID (UID) voor elke bezoeker aan te maken.

Een PHP Sessie Starten

Voordat er gebruikers informatie in een PHP sessie kan worden opgeslagen, moeten we eerst de sessie starten.

N.b.: De `session_start()` functie moet VOOR de `<html>` tag staan:

```
<?php session_start(); ?>
<html>
<body>
</body>
</html>
```

De bovenstaande code zal de gebruikers sessie registreren met de server, zal je toestaan informatie op te slaan in sessie variabelen en, zal een UID toewijzen aan de sessie.

Opslaan van een Sessie Variabele

De juiste manier om een sessie variabele op te slaan en uit te lezen is door de PHP `$_SESSION` variabele te gebruiken:

```
<?php
session_start();
// sla sessie data op
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//haal sessie data op
echo "Pagina Weergaves=". $_SESSION['views'];
?>
</body>
</html>
```

Weergave:

```
Pagina Weergaves=1
```

In het onderstaande voorbeeld maken we een eenvoudige pagina teller. De `isset()` functie bekijkt of de "views" variabele al bestaat. Als "views" al bestaat dan wordt de teller 1 verhoogd, als "views" niet bestaat wordt deze aangemaakt en op 1 gezet:

```
<?php
session_start();
if(isset($_SESSION['views']))
    $_SESSION['views']=$_SESSION['views']+1;
else
    $_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Vernietigen van een Sessie

Als je sessie data wilt wissen, kan je de `unset()` of de `session_destroy()` functie gebruiken. De `unset()` functie wordt vrij gebruikt voor een gespecificeerde variabele:

```
<?php
unset($_SESSION['views']);
?>
```

Je kan ook de gehele sessie vernietigen door de `session_destroy()` functie aan te roepen:

```
<?php
session_destroy();
?>
```

N.b.: `session_destroy()` zal de sessie resetten, alle sessie data gaat verloren.

PHP Verzenden van E-mail

Met PHP kan je een email sturen vanuit het script.

De PHP mail() Functie

De PHP mail() functie wordt gebruikt om email te verzenden vanuit een script.

Syntax

```
mail(aan, onderwerp, bericht, headers, parameters)
```

Parameter	Omschrijving
aan	Verplicht. Specificeerd de ontvanger/ontvangers van de email
onderwerp	Verplicht. Specificeerd het onderwerp van de email. N.b.: Deze parameter kan geen newline characters bevatten.
bericht	Verplicht. Het email bericht. Elke regel moet gescheiden worden met een linefeed, LF (\n). Regels zouden niet meer dan 70 characters moeten hebben.
headers	Optioneel. Specificeerd toevoegingen, zoals Van, Cc, en Bcc. Deze toevoegingen moeten gescheiden worden met een CRLF (\r\n)
parameters	Optional. Specificeerd een toegevoegde parameter voor het mail programma.

N.b.: Willen de mail functies beschikbaar zijn, zal dit wel in PHP geïnstaleerd moeten zijn.

PHP Eenvoudige E-Mail

De meest eenvoudige manier om email met PHP te verzenden, is het verzenden van tekst email.

In het onderstaande voorbeeld declareren we eerst de variabelen (\$aan, \$onderwerp, \$bericht, \$van, \$headers), daarna gebruiken we de variabelen in de mail() functie om een e-mail te verzenden:

```
<?php
$aan = iemand@voorbeeld.nl;
$onderwerp = "Test mail";
$bericht = "Hallo! Dit is een eenvoudig email bericht.";
$van= iemandanders@voorbeeld.nl;
$headers = "From: $van";
mail($aan,$onderwerp,$bericht,$headers);
echo "Mail Verzonden.";
?>
```

PHP Mail Formulier

Met PHP, kan je een feedback formulier om je site maken. Het onderstaande voorbeeld verzend een tekst bericht naar een gespecificeerd email adres:

```
<html>
<body>
<?php
if (isset($_REQUEST['email']))
//als email is ingevuld dan verzenden
{
//verzend email
$email = $_REQUEST['email'] ;
$subject = $_REQUEST['subject'] ;
$message = $_REQUEST['message'] ;
mail( iemand@voorbeeld.nl, "Subject: $subject",
$message, "From: $email" );
echo "bedankt voor de feedback";
}
else
//als "email" niet is ingevuld dan het formulier weergeve
{
echo "<form method='post' action='mailform.php'>
Email: <input name='email' type='text' /><br />
Onderwerp: <input name='subject' type='text' /><br />
Bericht:<br />
<textarea name='message' rows='15' cols='40'>
</textarea><br />
<input type='submit' />
</form>";
}
?>
</body>
</html>
```

Zo werkt het bovenstaande voorbeeld:

- Kijk eerst of het email veld is ingevuld
- zo niet (bijvoorbeeld als de pagina voor het eerst wordt bezocht); laat dan het HTML formulier zien
- zo ja (nadat het formulier is ingevuld); verzend de e-mail