PHP voor beginners

4 April 2008, voor Codequest.nl

De zin is toepasbaar in verschillende contexten: "Eens moet de eerste keer zijn". In dit artikel zal ik de beginnende webdevelopers op weg helpen.

Introductie

Dit artikel zal ingaan op hoe je jouw website meer functionaliteiten kunt geven door middel van PHP. Ik hoop het op zo een manier te kunnen uitleggen, dat het zèlf opzetten van oplossingen voortaan op een simpele manier zal verlopen. Hierbij denk ik dus niet aan "Hello World"-voorbeelden, maar aan situaties die je waarschijnlijk eerder nodig gaat hebben. Maar voordat we kunnen beginnen met PHP, is het handig als we weten wat PHP inhoud.

Wat is PHP?

PHP: Hypertext Preprocessor.

Hypertext Preprowattes?

Laten we bij het begin beginnen. Een versimpelde opzet van het internet, zou als volgt weergeven kunnen worden:

```
1. Jouw computer -> Internet -> Website -> Internet -> Jouw computer
```

Wanneer jij in je browser een website opzoekt, zal dit verzoek doorgestuurd worden naar de computer waarop deze website draait. Deze zal vervolgens de opgevraagde pagina terugsturen. Dit kun je zien als het opvragen van informatie van een bedrijf via de post; Jij stuurt een brief naar een bedrijf waarin je vraagt om meer informatie over iets. Zij sturen vervolgens die informatie terug.

Wanneer jouw website alleen bestaat uit HTML, zal er niets gebeuren in tussentijd. Oftewel, wanneer jij een brief stuurt naar een bedrijf waarin jij vraagt om informatie over een bepaald product, is er bij dit bedrijf niemand die hier naar kijkt. Je zal de basis informatie terug ontvangen omdat niemand weet dat je vroeg naar een specificiek product.

Met (o.a.) PHP kun je hier tussen gaan zitten. Je kunt dan (om maar bij het voorbeeld te blijven) de brief openenen, zoeken naar welk product gevraagd is en de juiste informatie terug sturen. PHP is dus geïnstalleerd op de computer waarop de website draait, dus is server-side.

Kort samengevat

Wanneer jouw website alleen bestaat uit HTML, zal deze pagina voor àlle bezoekers hetzelfde zijn. Wanneer je hier PHP in gebruikt, kan deze pagina afgestemt worden op de informatie die gevraagd wordt.

PHP ondersteuning

Wanneer je PHP wilt gebruiken, is het noodzakelijk dat het bedrijf dat jouw webhosting regelt ook PHP ondersteund. Vraag dit bij jouw provider alvorens PHP te gebruiken op uw website.

Wanneer u net begint met PHP is het misschien handiger om dit op uw eigen computer te installeren. U hoeft uw aanpassingen dan niet telkens te uploaden om de wijzigingen te bekijken. Wanneer u hier bekent mee bent kunt u dit natuurlijk zelf doen. Voor diegene die hier niet bekent mee zijn, zijn er kant-en-klaar pakketten zoals Wampserver.

Let op: in dit artikel zal ik voor de pagina's werken met de extensie .php. Schrik hier niet van. Bij de meeste webhosting bedrijven is het mogelijk jouw .html bestanden te hernoemen naar .php bestanden, zonder dat er ook maar iets veranderd.

PHP in de praktijk

Nu we weten wat PHP ongeveer inhoud, kunnen we kijken hoe dit in de praktijk zal werken. Een simpel voorbeeld hiervan is het weergeven van de naam van de bezoeker wanneer hij deze ingevoerd heeft. Stel je het volgende voor:

```
1.  <html>
2.      <head>
3.          <title>Testpagina</title>
4.      </head>
5.      <body>
6.          <h1>Welkom bezoeker</h1>
7.          <form action="welkom.php" method="post">
8.              <fieldset>
9.                  <legend>Voer uw naam in</legend>
10.                 <label>Naam:</label>
11.                 <input type="text" name="naam" />
12.                 <label>Doorgaan</label>
13.                 <input type="submit" value="Doorgaan!" />
14.             </fieldset>
15.         </form>
16.     </body>
17. </html>
```

Zoals je misschien bekend is, zal de pagina 'welkom.php' nu aangeroepen worden. Dit maakt nog niet dat er iets gebeurd met de informatie. Wel kan de meegestuurde informatie nu benaderd worden. In 'welkom.php' nemen we de volgende code op:

```
1.  <?php
2.      // Voor de oplettende kijker: PHP open je dus met <?php en sluit je
3.      // vervolgens weer met ?>
4.      echo "Welkom bij jouw eerste PHP-script, " . $_POST['naam'];
5.  ?>
```

Dat viel mee, toch? Jouw eerste PHP-script is gemaakt.
Dit ziet er misschien leuk uit, maar natuurlijk kun je met PHP veel leukere dingen doen...

Gebruikersinput

In het vorige voorbeeld heb ik laten zien hoe je een bezoeker zijn naam kan laten invullen, en hoe deze vervolgens te weergeven. Je hebt verschillende manieren om input te gebruiken binnen PHP. De meest voorkomende zal ik hieronder beschrijven.

Let op: het is leuk om op deze manier wat dingetjes heen en weer sturen. Echter moet je oppassen met hoe je onderstaande gebruikt. Deze zogenoemde variabelen kunnen door jou (of jouw website) ingevuld worden, maar ook door alle andere willekeurige bezoekers op jouw website. Later in dit artikel zal ik een klein beetje verder ingaan op de veiligheid van dit soort variabelen, maar mijn advies is je hier goed in te verdiepen alvorens je jouw website online gaat zetten.

POST-variabelen

In het vorige voorbeeld hebben we de POST-variabelen gezien. Stel, we willen dat iemand zijn naam en telefoonnummer in kan vullen. Hiervoor maken we het volgende formulier.

```
1. <!-- Let op: action="post"... we willen dit immers meegeven aan de post
   variabelen -->
2. <form action="post" method="verwerking.php">
3.     Naam: <input name="naam" type="text" />
4.     Telefoonnummer: <input name="telefoonnummer" type="text" />
5.     <input type="submit" value="Versturen!" />
6. </form>
```

Wanneer je dit formulier verstuurd, wordt de pagina verwerking.php opgeroepen. Een verschil met het normaal opvragen van een pagina, is dat er nu POST variabelen meegestuurd zijn, namelijk diegene uit het formulier dat we verzonden hebben. Deze variabelen krijgen de namen die je hebt opgegeven in het 'name'-attribuut van de 'input'-tag. We kunnen nu de volgende PHP code maken:

```
1. <?php
2.     // De variabelen 'naam' en 'telefoonnummer' zijn meegegeven
3.     // in de post variabelen. Deze zijn direct aanroepbaar door
4.     // middel van $_POST, op de volgende manier.
5.     // echo: we gaan iets weergeven
6.     echo "Uw naam is " . $_POST['naam'] . " en uw telefoonnummer is " .
   $_POST['telefoonnummer'];
7.     // $_POST['naam'] is datgene dat je achter Naam: hebt ingevuld
8.     // $_POST['telefoonnummer'] is datgene dat je achter
9.     //     Telefoonnummer: hebt ingevuld
10. ?>
```

GET-variabelen

Naast het versturen van variabelen via POST, kun je ook GET gebruiken. Deze zijn op de volgende manier te herkennen, je zult ze vast wel eens gezien hebben: http://www.jouwdomein.nl/index.php?id=1. Zoals je ziet wordt er achter de pagina die opgevraagd wordt, nog een extra waarde meegegeven, namelijk id=1. Dit is een GET-variabelle. Binnen PHP kun je deze als volgt gebruiken:

```
1. <?php
2.     // Pagina die opgeroepen is: http://www.jouwdomein.nl/index.php?id=1
3.     // De variabelle 'id' is nu meegegeven aan de GET-variabelen. Deze
4.     // zijn oproepbaar vanuit $_GET.
5.     // echo: we gaan iets weergeven
6.     echo "De pagina met het volgende ID is opgevraagd: " . $_GET['id'];
7. ?>
```

Het voordeel van deze variabelen, is dat de variabelen bij de zongenoemde URL horen. Wanneer je deze pagina's dus toevoegd aan de favorieten, zijn deze later met dezelfde variabelen weer op te vragen. Ook kun je deze variabelen rechtstreeks meegeven in een link.

```
1. <a href="/index.php?id=1">Linkje</a>
```

Natuurlijk zijn deze variabelen ook mee te geven via een formulier:

```
1. <!-- let op het action-attribuut binnen form -->
2. <form action="get" method="verwerking.php">
3.     ID: <input name="id" type="text" />
4. </form>
```

Cookies en sessies

Nu we wat verder gevorderd zijn in het gebruik van PHP, zullen we gelijk maar even wat termen gaan gebruiken. Op de vorige pagina hebben we kennis gemaakt met POST - en GET-variabelen. Deze waren op te roepen met $_POST en $_GET. Deze variabelen noemen we superglobals. Superglobals zijn variabelen die altijd en overal beschikbaar zijn, binnen jouw PHP code.

Je hebt verschillende superglobals met ieder hun eigen inhoud. De meest vergelijkbare met die van het vorige hoofdstuk zijn:

- $_COOKIE - De variabelen die vanuit een cookie meegestuurd worden
- $_SESSION - De variabelen die opgeslagen zijn in een sessie.

Je kunt deze superglobals op dezelfde manier gebruiken als de $_POST en $_GET superglobals. Echter wordt de inhoud hiervan niet meegegeven door middel van het opvragen van de pagina...

Cookies

Cookies zijn bestandjes die opgeslagen worden op de computer van de bezoeker van jouw website. Hierin kun je variabelen opslaan, met hieraan verbonden een geldigheid. De variabelen die opgeslagen worden, zijn dus beschikbaar totdat de geldigheid van een cookie verloopt, (let op:) óf totdat de gebruiker de cookie verwijderd. Omdat deze variabelen opgeslagen worden op de computer van de gebruiker, heeft deze natuurlijk ook de mogelijkheid deze weg te gooien. Ook is het

voor een bezoeker mogelijk cookies te blokkeren. Ook dan zal het opslaan van variabelen hierin natuurlijk niet werken.

Een simpel voorbeeld van het gebruik van cookies:

```
1. pagina1.php
2. <?php
3. // Een cookie met de naam 'test' en inhoud 'inhoud' opslaan:
4. // Je ziet hier het gebruik van de functie time(). Deze functie
5. // geeft het huidige tijdstip weer in seconden. Wanneer je
6. // dus time() + 3600 invult, zal deze cookie 3600 seconden
7. // (dus een uur) nadat de cookie is aangemaakt verlopen (verwijderd
8. // worden)
9. setcookie('test', 'inhoud', (time() + 3600));
10. ?>
11.
12. pagina2.php
13. <?php
14. echo "Inhoud cookie test: " . $_COOKIE['test']; // Op een tweede pagina is de
    variabelle weer beschikbaar.
15. ?>
```

Let op: Cookies zijn alleen beschikbaar binnen een domein. Je kunt dus niet op een website onder www.jouwdomein.nl een cookie instellen en vervolgens op www.mijndomein.nl dezelfde cookie uitlezen.

Sessies

Een andere methode om variabelen over meerdere pagina's op te slaan is het gebruik van sessies. Deze variabelen kunnen, in tegenstelling tot bij cookies, op de server (dus zegmaar op de website) opgeslagen worden. In de voorgaande zin moet je het woord 'kunnen' niet over het hoofd zien. Standaard werken sessies namelijk door middel van cookies, datgene dat ik hiervoor heb besproken.

Met deze methode worden de variabelen op de website opgeslagen, máár wordt er in een cookie een ID aangemaakt, dat verwijst naar deze variabelen. Op deze manier zijn ze terug te vinden. Als de bezoeker van jouw website dan dus de cookie weggooid, zal de sessie alsnog niet meer geldig zijn.

Als gezegd, dit staat standaard ingesteld. Dit kan eventueel uitgezet worden, maar hier zal ik in dit artikel niet op ingaan.

Zoals de naam al aangeeft: de informatie die opgeslagen wordt in een sessie, is ook alleen aanwezig gedurende een sessie. Wanneer jij gegevens hebt opgeslagen van een bezoeker en de bezoeker vervolgens zijn browser afsluit, zullen deze gegevens verloren gaan. Wanneer je wilt dat gegevens weer beschikbaar zijn wanneer een bezoeker terugkomt op jouw website, gebruik dan cookies.

Sessies kunnen op de volgende manier gebruikt worden:

```
1.  pagina1.php
2.  <?php
3.      // Start de sessie (geef aan dat je de sessie wilt gebruiken)
4.      session_start();
5.      $_SESSION['test'] = "inhoud"; // Maak een sessie variabele met naam 'test'
6.  ?>
7.
8.  pagina2.php
9.  <?php
10.     // Start de sessie (geef aan dat je de sessie wilt gebruiken)
11.     session_start();
12.     echo "Inhoud sessie test: " . $_SESSION['test']; // Op een tweede pagina is
    de variabelle weer beschikbaar.
13. ?>
```

Variabelen

In voorgaande hoofdstukken heb ik je laten zien hoe je informatie vanuit formulieren binnen jouw PHP code kunt gebruiken. Echter is dit niet de enige manier waarop je variabelen kunt gebruiken. In voorgaande hoofdstukken herkennen we iets gemeenschappelijks aan de verschillende soorten variabelen.

```
1.  <?php
2.      $_POST; // Begint met een dollar-teken
3.      $_GET; // Begint met een dollar-teken
4.      $_SESSION; // Begint met een dollar-teken
5.      $_COOKIE; // Begint met een dollar-teken
6.  // Hier ga ik opmerkingen over krijgen dus met deze alvast: de variabelen
7.  // beginnen inderdaad met een underscore. Dit is om deze variabelen zo min
8.  // mogelijk te laten botsen met variabelen die in de code gezet worden (lees
9.  // gauw verder).
10. ?>
```

Variabelen beginnen dus met een dollar-teken. Eigenlijk is het dus zo dat je een variabelle elke naam kan geven, zolang het maar met een dollar-teken begint. Deze variabelen kunnen we invullen op de volgende manier:

```
1.  <?php
2.  $variabelle = "een waarde";
3.  $variabelle_met_een_hele_lange_naam = "iets anders";
4.  $aantal = 1; // Nummers mogen buiten quotes (")
5.  ?>
```

De variabelen 'variabelle' en 'variabelle_met_een_hele_lange_naam' zijn in dit kleine stukje code dus allebei ingevuld. Deze kun je dus gebruiken in de rest van de code:

```
1.  <?php
2.  echo "Variabelle bevat " . $variabelle;
3.  echo     "Variabelle     met     een     hele     lange     naam     bevat     "     .
    $variabelle_met_een_hele_lange_naam;
4.  ?>
```

Je ziet dat dit nog redelijk overzichtelijk is. Echter, naarmate de applicaties groter en groter worden, zul je merken dat het overzicht een beetje zoek raakt wat variabelen betreft. Geef variabelen daarom altijd zinnige namen, bijvoorbeeld:

```
1. <?php
2. // FOUT:
3. $vage_variabelle_die_niet_werkt = 5;
4. $blaaa = "index.php";
5. // GOED:
6. $aantalLetters = 5; // Een aantal letters
7. $homepageAdres = "index.php"; // Het adres van de homepage
8. ?>
```

Datatypen

Elke variabelle binnen PHP heeft een type. Echter bestaat er binnen PHP type juggeling. Dit houdt in dat het niet uitmaakt van wat voor soort een variabelle is. Je kunt een nummer zonder problemen gebruiken als string (serie tekens) en andersom. In de meeste handleidingen wordt echter wel aangegeven wat voor typen er verwacht worden binnen functies. Hieronder vind je de meest voorkomende typen op een rijtje, met een voorbeeld van hoe deze te gebruiken.

String - een serie tekens, bijv "4ofdi34" of "%*gfs#&"

```
1. <?php
2.     // Aanmaken van een string
3.     $string = "dfs#$fda";
4. ?>
```

Int(eger) - een nummer

```
1. <?php
2.     // Aanmaken van een integer
3.     $int = 425343; // Zonder quotes dus
4. ?>
```

Array - een verzameling van variabelen, bijv ("4ofdi34", 3943) of (3443, 3423)

```
1. <?php
2.     // Aanmaken van een array
3.     $array = array("var1", 343, "var2");
4. ?>
```

Object - voor gevordenen, een instantie van een klasse.

```
1. <?php
2.     // Aanmaken van een object
3.     $object = new Exception();
4. ?>
```

Let op: variabelen die op bovenstaande manier gebruikt worden, zijn niét overal beschikbaar. Als je net met PHP begint, hoef je hier eigenlijk nog geen rekening mee te houden. Meestal zullen ze dan wel beschikbaar zijn. Loop je hier echter tegenaan, zoek dan even rond op 'PHP variable scope'.

Functies

Zo, we weten nu ongeveer hoe we om moeten gaan met variabelen. Toch worden variabelen natuurlijk pas leuk als we er daadwerkelijk wat mee kunnen. Hiervoor zijn er binnen PHP tal van functies die je kunt gebruiken. Een heeft een iets andere structuur dan een variabelle. Dit is ook logisch; een functie gebruik je immers om iets doen met een variabelle. Het volgende is een voorbeeld van een functie:

```
1.  <?php
2.      // We maken een variabelle. Snap je dit niet? Lees dan nog even het
3.      // vorige hoofdstuk.
4.      $var = "hello world";
5.
6.      // Nu gaan we hier iets mee doen
7.      echo ucfirst($var);
8.  ?>
```

Wacht eens! Voor ucfirst staat geen dollar-teken. Dit is dus geen variabelle. Zeker als beginner kun je er nu vanuit gaan dat het hier gaat om een functie. Wanneer je in het bovenstaande blokje klikt op 'ucfirst', zul je worden doorgestuurd naar de handleiding van PHP. Hier zien we dat dit inderdaad een functie is. Deze functie zal het eerste argument teruggeven als variabelle, waarin alle woorden met een hoofdletter ingevuld zullen zijn.

```
1.  <?php
2.      // We maken een variabelle. Snap je dit niet? Lees dan nog even het
3.      // vorige hoofdstuk.
4.      $var = "hello world";
5.
6.      // Nu gaan we hier iets mee doen
7.      echo ucfirst($var); // Geeft: 'Hello World' (met hoofdletters dus)
8.  ?>
```

Een functie bestaat uit argumenten (de variabelen die je meegeeft) en een zogenoemde return-value (de variabelle die teruggegeven wordt). Als we in de handleiding kijken naar substr (een functie waarmee je een deel van een string uit een string kunt halen), zien we dat deze functie tenminste 2 argumenten nodig heeft (de derde is optioneel). Deze geef je gescheiden door een komma mee. We gebruiken het derde argument hier ook in.

```
1.  <?php
2.  // We maken een variabelle
3.  $var = "Hello world";
4.
5.  // Hieronder zetten we een deel van de variabelle om in een nieuwe
6.  // variabelle. Voor de geïnteresseerden: Van $var, beginnen we vooraan
7.  // in de string (dus bij '0') en nemen we 5 tekens (dus '5').
8.  $hello = substr($var, 0, 5);
9.  // $hello zal nu gevuld zijn met de 'return-value' van substr, wat de
10. // eerste 5 tekens van 'Hello world' (vanuit $var) zal bevatten: 'Hello'
11. ?>
```

Dit ziet er schematisch als volgt uit:

```
1. $var                    ->                    'Hello                    world'
2. Voer   'substr'   uit   met   de   argumenten   $var|'Hello   world',   0   en   5
3.      Pak   5   tekens   vanaf   de   0de   positie   van   $var|'Hello   world'
4.      Retourneer                          het                          resultaat
5. Vang       het       geretourneerde       resultaat       op       in       $hello
6. $hello -> 'Hello'
```

Veiligheid

Met PHP zijn een hoop leuke dingen te doen. Echter zijn er ook dingen waar je goed rekening mee moet houden. In PHP wordt veelal gebruik gemaakt van input van de bezoekers van de site. Het nadeel hiervan is dat een bezoeker van alles kan invullen, dus óók dingen die je misschien niet verwacht. Het is daarom zaak álles te verwachten binnen jouw code. Hier een voorbeeld van hoe het goed fout kan gaan:

```php
1. <?php
2. // We willen aan de hand van de pagina die meegegeven is aan de URL bepalen
3. // welke PHP pagina er geladen moet worden.
4. // Voorbeeld: www.jouwdomein.nl/index.php?pagina=fotoboek
5. $pagina = $_GET['pagina'];
6.
7. // require_once is een functie waarmee je PHP-code uit andere bestanden kunt
8. // toevoegen aan de code van het bestand dat is opgeroepen.
9. require_once($pagina . ".php");
10. // De pagina fotoboek.php zal nu toegevoegd worden aan de code.
11. ?>
```

Bovenstaande ziet er onschuldig uit. Maar wat als de volgende pagina opgeroepen wordt: 'http://www.jouwdomein.nl/index.php?pagina=http://www.anderesite.nl/hack'.
Het volgende bestand zal dan meegegeven worden aan require_once en dus ook toegevoegd worden aan de code: 'http://www.anderesite.nl/hack.php'. Er zal dus code, die jij helemaal niet kent, toegevoegd worden aan jouw eigen code. Dit willen we natuurlijk niet!

Wat kunnen we hier tegen doen?

Bij het bovenstaande voorbeeld zou je het aantal mogelijkheden van pagina's die meegegeven kunnen worden, kunnen limiteren. Hier zijn meerdere manieren voor. Er is hier al een heleboel over te vinden op internet. Google bijvoorbeeld even wat rond op 'PHP security' of 'PHP beveiliging'.

Er zijn veel verschillende manieren waarop websites onveilig kunnen zijn. Mijn advies is dat je jezelf hier in verdiept om het moment dat je bepaalde functionaliteiten gaat maken.

PHP is leuk, maar pas goed op met alles dat je maakt. Zeker voordat je het online zet!

Hoe nu verder?

Oké. We weten hoe PHP gebruikt kan worden; we weten hoe variabelen gebruikt kunnen worden; we weten ongeveer hoe je functies gebruikt; hoe gaan we nu verder? Eén artikel is simpelweg te kort om alles in op te nemen. Wel kan ik je een feit toereiken: het leren van PHP ligt bij jezelf.

Doordat deze taal in de afgelopen jaren erg gegroeid is in populariteit, is er veel informatie beschikbaar. Het is dus aan jezelf om hier een beetje doorheen te bladeren. Zoek bijvoorbeeld eens op wat de volgende code doet:

```
1. <?php
2. $var = "Hoi bezoeker van deze website";
3. echo "Wat leuk: " . strrev($var);
4. ?>
```

Ik maak hierboven gebruik van een functie strrev. In het bovenstaande blokje code is het mogelijk op dit woordje te klikken. Je zal dan automatisch doorverwezen worden naar de handleiding van PHP. Hier zul je zien wat deze functie doet. Op heel veel websites is dit mogelijk. Is dit niet mogelijk? De handleiding van PHP is heet makkelijk te benaderen. Zie je een functie die je niet kent? Ga naar het volgende adres: 'http://www.php.net/functienaam' (bijvoorbeeld: http://www.php.net/strrev).

Een tweede dat je zou kunnen proberen, is het oplossen van quests op deze website. Deze quests kunnen over van alles gaan en dwingen je daarom een beetje zèlf op zoek te gaan naar antwoorden. Tevens kun je na het oplossen van een quest feedback verwachten, waar je weer veel van kunt leren. Naarmate je quests aan het oplossen bent, zul je ziet dat je steeds meer gedreven wordt binnen deze taal en ook steeds minder naar de handleiding hoeft te gaan, alvorens iets op te lossen.

Hiernaast nog een andere website, waar je veel kunt vinden over het gebruik van PHP, is de website PHP Freakz. Naast de artikelen en de wiki die je daar vindt, vind je hier een grote hoeveelheid (**doorzoekbare**) forum topics waarin veel vragen beantwoord worden waar andere PHP gebruikers mee zaten. Kijk maar eens rond op deze site.

Last but not least, is er natuurlijk nog Google. Met andere woorden: kijk gewoon goed rond en probeer bij onduidelijkheden ook op te zoeken wáárom iets zo is. Gebruik niet altijd alles dat voorgekauwd wordt, maar ga ook op zoek naar wáárom iets op een bepaalde manier gedaan moet worden. Zo kom je een heel eind en leer je er nog iets van.

Versies en ondersteuning?

Soms kom je wel eens wat tegen waarvoor je hoge versies of speciale extensions (extra uitbreidingen op de reguliere PHP code) nodig hebt. Je kunt opvragen welke versies en ondersteuningen je hebt door de functie phpinfo aan te roepen, dus:

```
1. <?php
2. phpinfo();
3. ?>
```

Conclusie

Er is ontzettend veel mogelijk met de scripttaal PHP. De inhoud van jouw website wordt al snel dynamischer, waardoor de bezoekers eerder terug neigen te komen: nieuwe content betekend nieuwe redenen om de site nog eens te bezoeken.

Dit artikel omvatten echter nog maar de simpele basis van deze taal. Wanneer je je verder verdiept in deze taal, zullen zich nog veel meer functionaliteiten en handigheden laten zien.

- Je kunt bestanden uitlezen en wegschrijven.
- Je kunt bestanden uploaden via een simpel HTML formulier.
- Je kunt databases aanroepen om gegevens op te slaan voor bijvoorbeeld een blog of een gastenboek.
- Je kunt teksten manipuleren om zo bijvoorbeeld BBCode toe te passen
- Etc., etc.

Houd goed in gedachte: Rome is ook niet in 1 dag gebouwd. Voor het leren van PHP moet je echt even goed gaan zitten. Begin simpel en breid je applicatie steeds verder uit. Op deze manier ontdek je steeds meer functionaliteiten en mogelijkheden die PHP biedt.

Hiernaast moet je rekening houden met het feit dat de formulieren die jij in kan vullen, ook door andere ingevuld kunnen worden. Houd ten alle tijden rekening met de veiligheid van jouw website. Ben je hier niet zeker van? Vraag het! Je kan beter een vraag teveel gesteld hebben, dan een website hebben die van alle kanten verziekt wordt.

Ik hoop dat het artikel zo duidelijk is; alle feedback is in ieder geval welkom. Rest mij niets anders dan je succes te wensen met het uitdiepen van PHP.

Succes!