

# PHP Hypertext Preprocessor

B. Grymonpon

9 mei 2001

# Hoofdstuk 1

## Een login-module

In dit hoofdstuk gaan we de beginselen van sessions nader bekijken. Om sessions te kunnen organiseren moeten we eerst en vooral een gebruiker kunnen laten inloggen (aan de hand van een gebruikersnaam en een paswoord) en vervolgens hem gedurende zijn werk kunnen identificeren zonder hem telkens opnieuw zijn gebruikersnaam en paswoord te laten intypen. Uiteindelijk moet de gebruiker kunnen uitloggen of moet, als hij te lang niets gedaan heeft, dus waarschijnlijk vergeten uitloggen is, het systeem hem kunnen uitloggen. De organisatie van dit alles is vrij eenvoudig, het is wel opletten geblazen omdat je geen enkele situatie uit het oog mag verliezen die de veiligheid van de gebruiker zou kunnen comprimeren!

### 1.1 Het systeem

Het systeem werkt als volgt: de gebruiker zal inloggen. Indien zijn gebruikersnaam of zijn paswoord fout zijn dan zal hij opnieuw de mogelijkheid krijgen om in te loggen. Zijn zijn gebruikersnaam en paswoord juist dan wordt de session gestart. De gebruiker krijgt een (unieke) cookie toegestuurd die we ook op de server bijhouden. De identificatie zal vanaf dan gebeuren aan de hand van die cookie, immers, bij de volgende pagina wordt die cookie meegestuurd naar de server en daar zullen we controleren of we effectief zo'n cookie hebben bijgehouden. Als dat het geval is leiden we daaruit af met welke gebruiker we bezig zijn en sturen we hem de (beveiligde) data door. Vervolgens zullen we (om de veiligheid te garanderen) de gebruiker een nieuwe cookie meesturen en de cookie op de server dus ook vernieuwen. Indien we zo geen cookie hebben op de server dan is er geen identificatie en dus zijn we bezig met iemand die niet ingelogd is (bvb. iemand die een oude cookie op zijn PC heeft staan). Indien er een beveiligde pagina wordt opgevraagd maar er is geen cookie of de cookie is foutief dan wordt met de header een niet-beveiligde pagina meegegeven (de login-pagina) waar de gebruiker dus zal terechtkomen. Indien de gebruiker uitlogt dan wordt de cookie zowel op de server als bij de client verwijderd.

### 1.2 De database

In de database hebben we dus vier velden nodig. Eerst en vooral de twee velden waarin we de gebruikersnaam en het paswoord bijhouden waaruit de data komt die de gebruiker nodig heeft om te kunnen inloggen. Verder hebben we uiteraard ook een veld nodig waarin we de cookie opslaan die we meesturen met die gebruiker en waaruit we later de gebruiker opnieuw zullen kunnen identificeren. Tenslotte heeft iedere user ook een id, om de MySQL-PHP-programmatie eenvoudiger te laten verlopen, zodat we niet telkens de volledig naam (string) moeten meesleuren met de data maar dat een uniek nummer (integer) voldoende is. Dat dit ook sneller is zal u (hopelijk) niet verwonderen... Een snelle blik op de database-velden geeft ons dus het volgende:

```
mysql> create table users(  
-> id integer unsigned not null auto_increment primary key,  
-> username varchar(32),  
-> password varchar(32),  
-> cookie varchar(32));
```

```
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> show columns from users;
```

```

+-----+-----+-----+-----+-----+-----+
| Field      | Type                | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(10) unsigned    |      | PRI | 0        | auto_increment |
| username   | varchar(32)         | YES  |     | NULL     |                |
| password   | varchar(32)         | YES  |     | NULL     |                |
| cookie     | varchar(32)         | YES  |     | NULL     |                |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

### 1.3 Inloggen

Om een gebruiker te laten inloggen geven we hem een form met twee velden en een submit-button. In het ene veld kan de gebruiker zijn login typen en in het ander zijn paswoord.

---

```

<html>
  <head>
    <title>Fout</title>
  </head>

  <body>
    <h1>Zeuslinks Login</h1>
    <?php echo $message."<br>" ?>
    <form method=POST action="secretpage.php">
      Login:<input name="login_username" size=20><br>
      Pass:<input type="password" name="login_password" size=20><br>
      <input type="submit" value="Inloggen"><br>
    </form>
  </body>
</html>

```

---

De login-form

Zoals je in de code kan zien zal de authenticatie gebeuren in secretpage.php.

---

```

<?php
require("config.inc.php");
require("database.inc.php");
require("login.inc.php");
?>
<html>
  <head>
    <title>De beveiligde pagina</title>
  </head>

  <body>
    <h1>Ok!</h1>
    U mag hier zijn!<BR>
    <a href="secretpage2.php">Doorklikken naar pagina2</a>
    <a href="secretpage.php?logout=1">Uitloggen</a>

  </body>
</html>

```

---

Een beveiligde pagina

Nu, secretpage is uiteraard een beveiligde pagina en bevat dus de beveiligde data. De authenticatie zelf zit hem in de requires die je boven ziet. De eerste twee requires (config.inc.php en database.inc.php) bevatten de data voor de connectie met de database, daarvoor verwijzen we u door naar de vorige PHP-les. Het is login.inc.php die dus de volledige authenticatie doet. Indien er een gebruikersnaam en een paswoord gegeven zijn (wat bij het inloggen het geval is) dan worden deze gecontroleerd. Dit gebeurt in de functie `checklogin()`. Eerst en vooral wordt het paswoord geëncrypteerd met de `crypt` functie, en vervolgens wordt dat geëncrypteerde paswoord vergeleken met het paswoord dat in de database zit. Zo hoeven we geen 'plain text'-paswoorden bij te houden in de database. Indien we een gebruiker hebben met die naam en dat paswoord in de database dan is alles OK, dus: we geven hem de unieke cookie die we ook opslaan in onze database om de gebruiker in het vervolg te kunnen identificeren. Alles is in orde want we hebben de file `login.inc.php` doorlopen zonder problemen en komen terecht bij de (beveiligde) data die in de file `secretpage.php` zit, deze pagina wordt dus gewoon getoond want de gebruiker is op een correcte wijze ingelogd. Indien de gebruiker een verkeerde combinatie van een gebruikersnaam en een paswoord meegeeft dan zal in de file `login.inc.php` blijken dat we zo geen gebruiker in de database hebben en dus gaan we met de `header("Location:")`-functie de gebruiker doorverwijzen naar een niet-beveiligde pagina en vervolgens stoppen zodat de beveiligde gegevens nooit bereikt worden.

---

```
<?php
// logincontrole voor bepaalde pagina's
srand(time() + 1235);

function check_login($username, $password){
    if(!$res = DoQuery("SELECT id,password FROM users WHERE username='$username'"))
        return -1;
    if (NumRows($res) != 1)
        return -1;

    // de username zit dus maar 1 keer... goed tot nu toe...

    $row = FetchArray($res);
    $salt = substr($row["password"],0,2);
    $crypted_password = crypt($password,$salt);
    // echo "Check: $salt -- $crypted_password -- ".$row["password"];
    if ($crypted_password == $row["password"])
        return $row["id"];
    else
        return -1;
}

function set_new_cookie($uid){
    // een nieuw cookie maken, zetten bij de gebruiker
    // en aanpassen in de database...
    $cookie = uniqid("Zeus".$uid,true);
    if(!$res = DoQuery("UPDATE users SET cookie='$cookie' WHERE id='$uid'"))
        return false;
    if(!setcookie("login_cookie",$cookie))
        return false;
    return $cookie;
}

function get_uid_from_cookie($cookie){
    if(!$res = DoQuery("SELECT id,cookie FROM users WHERE cookie='$cookie'"))
        return false;
    // hmmm, goede query, nog eens het resultaat controleren...
    $row = FetchArray($res);
    switch(NumRows($res)){
    case 0:
```

```
// niemand te vinden met $cookie als cookie...
return false;
break;
case 1:
    return $row["id"];
    break;
default:
    // helaba? Wat is dat hier? LINKE BOEL! En we vertrouwen niets...
    setcookie("login_cookie","");
    DoQuery("UPDATE users SET cookie='' WHERE cookie='$cookie'");
    return false;
}
}

function logout($uid){
    DoQuery("UPDATE users SET cookie='' WHERE id='$uid'");
}

// Ok, hier begint het allemaal
if(!isset($login_cookie)){
    // Geen cookie! Hmm, dan moeten we kijken naar een username
    // en een password...
    if(empty($login_password) || empty($login_username)){
        $message=urlencode("Gelieve een username en password in te geven!");
        header("Location: wrong_login.php?message=$message");
        die();
    }
    if(-1 == ($uid = check_login($login_username,$login_password))){
        $message=urlencode("Username en/of password verkeerd!");
        header("Location: wrong_login.php?message=$message");
        die();
    }
    // blijkbaar hebben we een uid kunnen halen uit de username/password
} else {
    // aha, een cookie!
    if(!$uid = get_uid_from_cookie($login_cookie)){
        setcookie("login_cookie","");
        $message=urlencode("Er werd een verkeerde cookie aangetroffen!");
        header("Location: wrong_login.php?message=$message");
        die();
    }
}

// Wanneer we hier raken moet er ofwel een username/password paar
// correct zijn, ofwel moet er een correcte cookie gevonden zijn...
// Snel een nieuwe cookie bij de gebruiker zetten, en dan kunnen we
// beginnen uitvoer genereren en is ons werk gedaan...
if (empty($logout)){
    set_new_cookie($uid);
} else {
    logout($uid);
    setcookie("login_cookie","");
    header("Location: logout.html");
}
// einde van de module
?>
```

U had graag een kort woordje uitleg over de `crypt()`-functie en de `uniqid()`-functie? Wel, here we go... De `crypt()`-functie zal het paswoord zoals gezegd gaan encrypteren. Dit is een encryptie in 1 richting, dus het zal niet mogelijk zijn van datzelfde paswoord later opnieuw te gaan decrypteren. De salt die daarbij gebruikt wordt is niet meer dan de random-portie waarmee je paswoord geëncrypteerd is. Deze salt (2 karakters) wordt steeds voor je geëncrypteerde paswoord geplaatst. Het ingegeven paswoord moet uiteraard steeds geëncrypteerd worden met dezelfde salt want anders zou je geëncrypteerd stuk niet hetzelfde zijn en dus zou de gebruiker niet kunnen inloggen, vandaar dus ook dat we de salt eerst uit de database gaan halen en meegeven met `crypt()`. De `uniqid()`-functie doet eigenlijk niets meer dan een unieke key gaan creëren, wat dus uitermate geschikt is om voor onze cookie-key te gebruiken. In de code treffen we ook de `urlencode()`-functie aan. Dit is een handige functie die we kunnen gebruiken om zonder problemen data mee te geven in onze URL. Deze zet immers alle niet-alphanumerieke karakters om naar hun HTML-equivalent. Zo wordt een spatie bijvoorbeeld vervangen door `%20` en kunnen we op die manier heel eenvoudig een zin meegeven in onze URL. Voor meer informatie over al deze functies verwijzen we u door naar <http://www.php.net/manual>.

## 1.4 Identificatie

Indien er geen gebruikersnaam en paswoord zijn meegegeven dan zal dus de cookie gecontroleerd worden, indien er geen cookie is of de indien de cookie nergens voorkomt in onze database, of indien de cookie meer dan 1 keer voorkomt in de database (!!!) dan is de gebruiker niet ingelogd of kunnen we hem niet onderscheiden van een andere gebruiker en wordt hij dus opnieuw met de `header()`-functie doorverwezen naar de login-pagina. Indien er een cookie was dan zal die cookie eerst verwijderd worden. Indien de cookie 1 keer voorkomt dan is de gebruiker ingelogd en kunnen we aan de hand van de database dus zien met wie we te doen hebben. We geven hem dan eerst een nieuwe cookie en passen deze aan in de database. Vervolgens kan de gebruiker dan lustig verder surfen op de beveiligde pagina's.

## 1.5 Uitloggen

Indien de gebruiker de logout-link heeft aangeklikt dan verwijderen we de cookie uit de database, we verwijderen hem van de computer van de gebruiker en we verwijzen hem door naar een niet-beveiligde pagina. Dit kan vanop bijvoorbeeld de volgende beveiligde pagina gedaan worden:

---

```
<?php
require("config.inc.php");
require("database.inc.php");
require("login.inc.php");
?>
<html>
  <head>
    <title>De beveiligde pagina</title>
  </head>

  <body>
    <h1>U klikte!</h1>
    En hier ook<a href="secretpage.php">Terugklikken</a>
    <a href="secretpage2.php?logout=1">Uitloggen</a>

  </body>
</html>
```

---

Een beveiligde pagina met logout-link

---

```
<html>
  <head>
```

```
<title>Correct uitgelogd</title>
</head>

<body>
  <h1>U bent correct uitgelogd</h1>
  <a href="secretpage.php">Proberen de geheime pagina te lezen</a>

</body>
</html>
```

---

De niet beveiligde logout-pagina

## 1.6 Optimalisatie

### 1.6.1 Code-optimalisatie

Er zijn twee zaken aan onze bovenstaande code die beter kunnen, waarvan 1 eigenlijk wel als een fout kan bestempeld worden. Ten eerst moet als de gebruiker uitlogt php volledig de authenticatie-file doorlopen. Dit is eenvoudig op te lossen door de `if`-conditie anders te structureren. De fout in de code is eigenlijk dat als zowel een gebruikersnaam en een paswoord als een cookie op de server terechtkomen, de cookie voorrang krijgt. Ook dit is eenvoudig op te lossen door de `if-lus` anders te structureren, maar we laten dit met plezier aan u over ;-).

### 1.6.2 Autologout

In ons systeem is het zo dat de gebruiker, eens ingelogd, altijd ingelogd blijft. Dit is niet zo'n goed idee want veelal laten mensen hun PC even alleen om bijvoorbeeld een koffie te gaan halen. Als er dan iemand anders aan die PC gaat zitten dan zal die zomaar op die pagina's kunnen verder surfen, ookal heeft die daar eigenlijk geen toegang toe. Een oplossing hiervoor is in de database bvb. een veld `last_action` bijhouden van het type `datetime`. Daar kunnen we dan telkens de gebruiker een nieuwe cookie gekregen heeft de tijd aanpassen en de waarde van de huidige tijd geven. Telkens voor het moment dat de nieuwe cookie gegeven wordt kan dat veld opgehaald worden en vergeleken worden met de huidige tijd. Als het verschil tussen die 2 dan groter is dan een bepaalde waarde, bvb. 5 minuten, dan kunnen we de cookie verwijderen en de gebruiker doorverwijzen naar een niet-beveiligde pagina, waar hij opnieuw zal moeten inloggen.

### 1.6.3 Sessions

Een andere optimalisatie, om tot het echte session-management te komen, is gewoonweg meer data meegeven met dit alles. Zo kunnen we bvb bijhouden van welke pagina een gebruiker komt en naar welke pagina hij gaat. We kunnen dit dan bijhouden in logs om informatie in te winnen over ons systeem en het surfgedrag van onze gebruikers om zo tot verbeteringen te komen in de structuur van onze website en onze code.

### 1.6.4 Sessions in PHP4

Vanaf PHP4 ondersteunt PHP zelf ook sessions. Dit gebeurt door een Session IDentifier of SID. Deze variabele is vergelijkbaar met de key (cookie) die we hierboven hebben gebruikt in onze eigen implementatie. Die identifier werkt samen met de functie `session_register()`. Deze functie zorgt ervoor dat variabelen gedefinieerd zijn en hun waarde behouden doorheen een sessie. Als deze is afgelopen dan verdwijnen die variabelen. `session_register()` moet voorkomen in de PHP-code voor er enige uitvoer is want het is gebaseerd op een systeem van cookies, alles wordt dus met de headers meegestuurd. De beste info over deze sessions is te vinden op <http://www.php.net/manual>, beter kunnen we hem zelf niet geven. . .

## Hoofdstuk 2

# Extra's in PHP

In dit hoofdstuk gaan we enkele toevoegingen zien die PHP heeft. Deze extra modules moeten echter geïnstalleerd worden bij de installatie van PHP dus is het beter dat je eerst controleert met `phpinfo()` als een bepaalde module beschikbaar is op je server.

We gaan enkele voorbeelden zien hoe we met PHP grafische files kunnen aanmaken, hoe we XML documenten kunnen verwerken of hoe we PDF documenten kunnen aanmaken en doorsturen naar de browser.

### 2.1 Tekeningen maken en bewerken

Door middel van de GD Library kan PHP grafische bestanden lezen, schrijven en bewerken. Dit houdt in dat we een nieuw grafisch bestand kunnen aanmaken of een bestaand nemen, daar veranderingen aan aanbrengen en uiteindelijk dit bestand terug opslaan of doorsturen naar de browser.

Dit kan bijvoorbeeld handig zijn om tellers dynamisch aan te passen, om balkjes die bepaalde verdelingen weergeven dynamisch op te bouwen, om wegenkaarten op te bouwen en de gebruikers de mogelijkheid te bieden om in- en uit te zoomen, . . . Er zijn veel mogelijkheden waar dit nuttig kan zijn.

We zien ook hoe we een grafisch bestand in een form kunnen inbrengen, en dan de waarden terugkrijgen waar de persoon precies geklikt heeft.

#### 2.1.1 Een overzicht van het gebruik

Wanneer we grafische bestanden gebruiken in HTML-documenten gebeurt dit meestal met de volgende constructie:

```
<IMG SRC="images/voorbeeld.gif" ALT="Een voorbeeld" HEIGHT=20 WIDTH=30>
```

Er kunnen nog extra parameters meegegeven worden in de IMG-tag, maar deze hebben we hier niet nodig. Wanneer we nu echter een tekening gaan gebruiken die door een PHP-script opgebouwd werd moeten we volgende constructie gebruiken:

```
<IMG SRC="scrips/voorbeeld.php" . . .>
```

We kunnen terug extra parameters meegeven in de IMG-tag die de hoogte en de breedte gaan bepalen.

Een handig gebruik van dynamische tekeningen is bij het dynamisch maken van wegenkaarten, weerkaarten en dergelijke. Hiervoor kunnen we in een form een grafisch bestand invoegen en wanneer de gebruiker dan klikt op de tekening krijgt het script de x- en y-waarden terug waarmee we dan verder kunnen werken.

Op deze toepassing gaan we echter niet verder in omdat hierbij de GD bibliotheek niet gebruikt wordt.

Een kleine opmerking nog over de GD bibliotheek. In ons voorbeeld werd eerst een gif-bestand aangehaald. Wegens copyrights kan de GD bibliotheek echter niet verdeeld worden met de mogelijkheid om gif-bestanden te maken. De GD bibliotheek gebruikt echter het png-formaat om bestanden te gaan bewerken. Oudere versies van de bibliotheek kunnen echter wel gif-bestanden aanmaken. Aangezien iedere moderne browser echter perfect overweg kan met png-bestanden kan dit geen probleem vormen.

Let er echter op dat je best eerst controleert welk formaat ondersteund wordt door je versie van GD.

### 2.1.2 Een overzicht van het gebruik

Wanneer we een tekening naar de browser sturen, moeten we dit ook duidelijk maken aan de browser. Dit kunnen we door in de header extra informatie mee te geven die de inhoud beschrijft van de data die volgt na de header. Normaal is die data een HTML-bestand, maar wanneer we een tekening versturen is dit gewoon de inhoud van het grafisch bestand. We kunnen de browser vertellen dat we een grafisch bestand gaan doorsturen met de volgende header-constructie:

```
header("Content-type: image/gif");
```

In plaats van “gif” kunnen we natuurlijk ook “jpeg”, “png” of andere formaten meegeven. Wanneer we werken met de GD-bibliotheek zal dit dus “png” worden.

Wanneer we een grafisch bestand openen of aanmaken in PHP, dan kunnen we er verder naar verwijzen via een “handle”, net zoals bij database connecties of bestanden. Het is eigenlijk gewoon een geheel getal dat meegegeven wordt aan iedere functie die moet werken op de gevraagde tekening.

Een voorbeeld hoe we een grafisch bestand maken dat een lijn van linksboven naar rechtsonder tekent.

---

```
<?php
Header("Content-type: image/png");
srand(time(null));
$im = imagecreate(200,200);
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
ImageLine($im,0,0,200,200,$black);
ImagePng($im);
ImageDestroy($im);
?>
```

---

Een kort PHP-script dat een lijn tekent

In dit voorbeeld zien we het gebruik van enkele veelgebruikte functies. Eerst en vooral maken we een nieuw grafisch bestand aan met de functie `ImageCreate()`. Deze functie neemt 2 parameters, de hoogte en breedte van de tekening in pixels. Let erop dat deze functie geen bestaand bestand inleest, maar een leeg bestand aanmaakt in het geheugen. De functie `ImageCreate()` geeft een file-handle terug. Wanneer het niet gelukt is om een grafisch bestand aan te maken krijgen we een “false” terug.

Wanneer de tekening niet langer nodig is in het script (en dus enkel maar plaats inneemt) kunnen we die verwijderen met `ImageDestroy()`. De enige nodige parameter is de handle van de tekening die we wensen te vernietigen.

Wanneer we echter bestanden willen lezen en schrijven kunnen we hiervoor de functies `ImageCreateFromGif()` en `ImageGif()` gebruiken. De eerste functie neemt slechts 1 parameter, nl de bestandsnaam van het gif-bestand. We krijgen een handle-terug, en kunnen verder werken met de tekening alsof we gewoon een lege tekening hadden aangemaakt met `ImageCreate()`. `ImageGif()` schrijft de gemaakte tekening weg in gif-formaat naar een bestandsnaam. De eerste parameter is de handle van de tekening. De tweede parameter is de bestandsnaam. Wanneer deze echter niet meegegeven wordt, gaat PHP de tekening gewoon doorsturen naar de browser.

Enkele andere handige functies zijn de volgende:

**GetImageSize(filenaam, [extenden\_jpeg\_info ])** Met deze functie krijgen we een array terug met enkele waarden in. Op plaats 0 vinden we de breedte van de tekening. Element 1 bevat de hoogte. Element 3 geeft aan als het formaat aan. (1=gif, 2=jpg en 3=png). Op de vierde plaats vinden we een string met als inhoud “WIDTH=xx HEIGHT=yy” die we onmiddellijk kunnen gebruiken in een IMG-tag. Let erop dat deze functie enkel gebruikt kan worden op tekeningen die opgeslagen zijn op de harde schijf.

**ImageSX(handle)** Deze functie geeft de breedte terug van een tekening in het geheugen. Deze tekening mag ofwel van de harddisk komen, of mag manueel aangemaakt zijn.

**ImageSY(handle)**— Identiek aan `ImageSX()`, maar geeft de hoogte terug in plaats van de breedte.

### 2.1.3 Kleuren

Wanneer we kleuren gebruiken op bestanden kunnen we slechts een beperkt aantal kleuren gebruiken binnen 1 grafisch bestand. Dit aantal is afhankelijk van het formaat van de tekening en wordt in vaktermen de “palet” genoemd. Voor gif-bestanden is dit bijvoorbeeld 256 kleuren. Wanneer we dus kleuren gebruiken moeten we erop letten dat we niet teveel kleuren gebruiken. Er bestaan functies om een kleur op te zoeken in een tabel, om een kleur te benaderen of om een nieuwe kleur aan te maken. We kunnen ook kleuren gaan vervangen in de palette, waardoor alle pixel met die bepaalde kleurwaarde dan een andere kleur gaan krijgen.

Alle kleurbewerkingen verlopen met de rgb-kleurindex die ook gebruikt wordt binnen HTML-documenten. Iedere index kan een waarde aannemen tussen 0 en 256.

**ImageColorAllocate(handle, r, g, b)** Deze functie maakt binnen de tekening een nieuwe kleur aan. Wanneer deze functie lukt krijgen we een kleur-identificer terug (het nummer van het kleur in de palette). De eerste maal dat deze functie aangeroepen wordt, wordt de achtergrondkleur ook ingesteld op de meegegeven waarde. In bovenstaand voorbeeld gaan we dus de achtergrondkleur op zwart instellen.

**ImageColorAt(handle, x, y)** Deze functie geeft de index binnen de palette terug van het kleur van de pixel op de tekening.

**ImageColorClosest(handle, r, g, b)** Deze functie maakt geen nieuwe kleur aan, maar gaat binnen de reeds bestaande palette de beste benadering zoeken voor het opgegeven kleur. De waarde van het dichtste kleur wordt teruggegeven.

**ImageColorExact(handle, r, g, b)** Deze functie zoekt binnen een palette naar een bepaald kleur. Wanneer het reeds aanwezig is, krijgen we de waarde ervan terug, anders krijgen we -1 terug.

**ImageColorResolve(handle, r, g, b)** Deze functie is een samenstelling van de vorige functies. Eerst wordt gezocht binnen de palette naar het exacte kleur. Wanneer dit niet gevonden wordt proberen we een nieuwe kleur aan te maken met de opgegeven kleurwaarden. Wanneer ook dit niet lukt gaan we met ImageColorClosest gaan zoeken naar de beste benadering voor het gevraagde kleur. Deze functie geeft dus *altijd* een kleur terug.

**ImageColorSet(handle, index, r, g, b)** Deze functie gaat het kleur met index “index” gaan vervangen door de opgegeven kleurwaarden.

**ImageColorsForIndex(handle, index)** Deze functie geeft een array terug met de 3 waarden voor rood, groen en blauw terug van het kleur met de gegeven index.

**ImageColorsTotal(handle)** Deze functie geeft het aantal reeds gebruikte kleuren terug in de palette van de tekening.

**ImageColorTransparent(handle, [index ])** Deze functie zet een bepaalde index op transparant. Dit kleur wordt dan niet weergegeven maar vervangen door de achtergrondkleur van het huidige venster in de browser. Wanneer we geen waarde opgeven krijgen we de index terug van het huidige transparante kleur. Wanneer we de transparantie willen uitschakelen geven we een index van “-1” mee.

### 2.1.4 Geometrische figuren

Om lijnen, cirkels en veelhoeken te tekenen kunnen we enkele functies gebruiken. We geven hieronder een overzicht van de meestgebruikte functies. Let erop dat bij het coördinatensysteem van PHP de oorsprong linksboven ligt en niet linksonder zoals we gewoon zijn.

**ImageLine(handle, bx, by, ex, ey, kleur)** Deze functie tekent een lijn vanaf de begincoördinaten (bx,by) tot aan de eindcoördinaten (ex,ey) in het opgegeven kleur.

In het bovenstaande voorbeeld gebruikten we deze functie om de lijn te trekken op onze tekening.

**ImageDashedLine(handle, bx, by, ex, ey, kleur)** Deze functie is identiek aan ImageLine() maar tekent een stippellijn.

**ImageArc(handle, cx, cy, breedte, hoogte, start, eind, kleur)** Deze functie tekent een deel van een ellips. De ellips krijgt als centrum (cx,cy) en een opgegeven hoogte en breedte. De start en eindwaarden zijn graden (geen radialen!). Om een volledige ellips te tekenen gebruiken we als beginwaarde nul en eindwaarde 360.

---

```
<?php
Header("Content-type: image/png");
srand(time(null));
$im = imagecreate(400,100);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
for ($i=1; $i < 10; $i++)
    ImageArc($im,200,0,380 / $i,180 / $i,5,175,$white);
ImagePng($im);
ImageDestroy($im);
?>
```

---

Het gebruik van ImageArc()

**ImageRectangle(handle, x1, y1, x2, y2, kleur)** Deze functie tekent een rechthoek met hoeken in (x1,y1) en (x2,y2).

**ImagePolygon(handle, punten, aantal, kleur)** Deze functie tekent een veelhoek. De array "punten" bevat telkens opeenvolgende x en y waarden. Punten[0] = x0, punten[1]=y0,..Het aantal punten wordt weergegeven in "aantal".

**ImageFilledRectangle()** Deze functie is identiek aan ImageRectangle, maar vult de rechthoek op met het opgegeven kleur.

**ImageFilledPolygon()** Deze functie is identiek aan ImagePolygon, maar vult de veelhoek op met het opgegeven kleur.

---

```
<?php
Header("Content-type: image/png");
srand(time(null));
$im = imagecreate(200,200);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
for($i=0; $i <20; $i++){
    $punten[$i] = rand(0,200);
}
ImageFilledPolygon($im,$punten,10,$white);
ImagePng($im);
ImageDestroy($im);
?>
```

---

Het gebruik van ImageFilledPolygon()

**ImageFill(handle, x, y, kleur)** Deze functie gaat een flood-fill toepassen op de tekening vanaf het opgegeven punt. De tekening wordt gevuld met de opgegeven kleur.

**ImageFillToBorder(handle, x, y, boordkleur, kleur)** Deze functie gaat een flood-fill toepassen op de tekening vanaf het opgegeven punt. De tekening wordt gevuld met de opgegeven kleur tot aan een boord met het opgegeven kleur.

### 2.1.5 Tekst

We kunnen ook tekst zetten op een tekening. PHP biedt heel wat functies om dit te doen. We zien hieronder enkele simpele functies. Voor meer uitgebreide uitleg verwijzen we naar de manual van PHP.

Om lettertypes te gebruiken kunnen we ofwel de 5 ingebouwde fonts gebruiken (aangeduid door de nummers 1 tot 5), ofwel zelf fonts laden en gebruiken. Om de TTF-functies te gebruiken moeten we bijkomend ook nog de FreeType-bibliotheek installeren.

**ImageString(handle, font, x, y, tekst, kleur)** De functie gaat een bepaalde tekst gaan uitschrijven vanaf een opgegeven punt. De kleur en het lettertype kunnen meegegeven worden.

**ImageFontHeight(font)** Deze functie geeft de hoogte van een font terug.

**ImageFontWidth(font)** Deze functie geeft de breedte van een font terug. Deze functie werkt enkel met een mono-spaced font (zoals de 5 ingebouwde fonts).

**ImageLoadFont(bestand)** Deze functie gaat een fontfile (mono-spaced) gaan laden en beschikbaar maken doorheen het script. Deze functie geeft een nummer terug (dan boven de 5 ligt) om het font verder te kunnen gebruiken.

**ImageTTFText(handle, grootte, hoek, x, y, kleur, TTFfile, tekst)** Deze functie gaat een tekst uitschrijven met een opgegeven TTF-font. De hoek en startpositie kan meegegeven worden.

**ImageTTFBox(grootte, hoek, TTFfile, tekst)** Deze functie geeft een array terug met de hoekpunten van een bepaalde tekst. Voor meer informatie hierover verwijzen we naar de manual van PHP.

### 2.1.6 Een logo-generator

---

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
  <head>
    <title>PHP is cool!</title>
  </head>

  <body bgcolor="#FFFFFF">
    <h1>PHP is cool!</h1>
    <IMG SRC="php_is_cool.php?s=10&text=PHP+is+Cool!"><BR>
    <IMG SRC="php_is_cool.php?s=20&text=PHP+is+Cool!"><BR>
    <IMG SRC="php_is_cool.php?s=40&text=Zeus+is+Cool!"><BR>
  </body>
</html>
```

---

Hoe een PHP-tekening in te voegen

---

```
<?php
Header("Content-type: image/png");
if(!isset($s)) $s=11;
$size = imagegetttfbbox($s,0,"HURRYUP.TTF",$text);
$dx = abs($size[2]-$size[0]);
$dy = abs($size[5]-$size[3]);
$ypad=9;
$ypad=9;
$im = imagecreate($dx+$ypad,$dy+$ypad);
$blue = ImageColorAllocate($im, 0x2c,0x6D,0xAF);
$black = ImageColorAllocate($im, 0,0,0);
$white = ImageColorAllocate($im, 255,255,255);
ImageRectangle($im,0,0,$dx+$ypad-1,$dy+$ypad-1,$black);
```

```

ImageRectangle($im,0,0,$dx+$xpad,$dy+$ypad,$white);
ImageTTFText($im, $s, 0, (int)($xpad/2)+1, $dy+(int)($ypad/2), $black, "HURRYUP.TTF", $text);
ImageTTFText($im, $s, 0, (int)($xpad/2), $dy+(int)($ypad/2)-1, $white, "HURRYUP.TTF", $text);
ImagePng($im);
ImageDestroy($im);
?>

```

---

Een logo ontwerpen met een TTF

## 2.2 XML

*Opmerking!* De uitleg die hier gegeven wordt werd overgenomen van [www.phpbuilder.com](http://www.phpbuilder.com). Alle copyrights zijn dan ook eigendom van [phpbuilder.com](http://www.phpbuilder.com).

XML is een dataformaat dat meer en meer als standaard wordt gebruikt om een gegevens uit te wisselen. Het heeft ongeveer een structuur zoals een HTML-document, met begin en eind-tags. Tussen de verschillende tags staat er data. HTML gebruikt tags echter om de structuur en opmaak van een pagina op te slaan terwijl XML de tags gebruikt om gegevens omtrend de data op te slaan.

Als voorbeeld gebruiken we de XML-file van slashdot. Deze XML-file is volledig volgens de standaarden opgebouwd en dus perfect voor experimentele doeleinden.

De XML-ondersteuning die PHP biedt is vrij simpel. Je kan een functie definiëren die elke open-tag behandelt, een functie voor de sluittags en een functie voor de data tussen de tags. Wanneer deze 3 functies opgebouwd zijn openen we de file en gaan die door de parser gaan verwerken. Al het werk wordt verder gedaan door de parser, die de gemaakte functies oproept wanneer het nodig is.

### 2.2.1 De structuur van slashdot.xml

De structuur van het slashdot.xml-document is een mooi voorbeeld van de opbouw van een XML-document. We beginnen telkens met een lijntje dat de versie van de XML weergeeft die gebruikt wordt. Verder krijgen we telkens begin en een eind-tags met data tussen deze tags. Tussen verschillende tag mogen er natuurlijk terug nieuwe tags staan. We kunnen een XML-document goed gaan vergelijken met een tabel in een database.

---

```

<?xml version="1.0" encoding="ISO-8859-1"?><backslash
xmlns:backslash="http://slashdot.org/backslash.dtd">

  <story>
    <title>How To Handle A Killer Asteroid</title>
    <url>http://slashdot.org/article.pl?sid=01/05/08/011209</url>
    <time>2001-05-08 07:55:13</time>
    <author>timothy</author>
    <department>liv-tyler's-turn</department>
    <topic>space</topic>
    <comments>81</comments>
    <section>articles</section>
    <image>topicspace.gif</image>
  </story>
...
  <story>
    <title>Ask an Attorney About Open Source Licensing</title>
    <url>http://slashdot.org/article.pl?sid=01/05/06/1823243</url>
    <time>2001-05-07 17:00:18</time>
    <author>Roblimo</author>
    <department>expert-advice-from-the-experts</department>
    <topic>doj</topic>
    <comments>256</comments>
    <section>interviews</section>

```

```

    <image>topicdoj.jpg</image>
</story>

<story>
  <title>On Starting a Successful ISP?</title>
  <url>http://slashdot.org/article.pl?sid=01/05/07/0653215</url>
  <time>2001-05-07 16:41:46</time>
  <author>Cliff</author>
  <department>born-in-a-shrinking-market</department>
  <topic>tech</topic>
  <comments>313</comments>
  <section>askslashdot</section>
  <image>topictech2.jpg</image>
</story>

</backslash>

```

---

Het XML-document

De bedoeling van dit voorbeeld is het maken van een parser voor dit document.

## 2.2.2 Het volledige programma

We bekijken het programma eerst volledig:

---

```

<html>
  <head>
    <title>
      Slashdot door onszelf...
    </title>
  </head>
  <body bgcolor="#FFFFFF">

<?php

// de tags die belangrijk zijn voor ons...
$open_tags = array('STORY' => '<STORY>',
                  'TITLE' => '<TITLE>',
                  'URL' => '<URL>');

$close_tags = array('</STORY>',
                  '</TITLE>',
                  '</URL>');

// deze functie zorgt voor de verwerking van de openings-tags
// We slaan de tag-waarde op in een globale variabele,
// Enkel wanneer we een "story"-tag hebben gaan we iets uitschrijven
function startElement($parser, $name, $attrs='') {
  global $open_tags, $temp, $current_tag;
  $current_tag = $name;
  if ($format = $open_tags[$name]) {
    switch($name) {
      case 'STORY':
        echo 'New Story: ';
        break;
      default:
        break;
    }
  }
}

```

```

    }
  }
}

// Deze functie behandelt de de sluit-tags.
// Wanneer we een sluittag van "story" tegenkomen gaan we
// alles uitschrijven, en ons klaarmaken voor een nieuwe
// aanvoer van data.

function endElement($parser, $name, $attrs='') {
  global $close_tags, $temp, $current_tag;
  if ($format = $close_tags[$name]) {
    switch($name) {
      case 'STORY':
        return_page($temp);
        $temp = '';
        break;
      default:
        break;
    }
  }
}

// Deze functie behandelt de data tussen de
// verschillende elementen.
// de variabele $data heeft de waarde van de data.
function characterData($parser, $data) {
  global $current_tag, $temp, $catID;
  switch($current_tag) {
    case 'TITLE':
      $temp['title'] = $data;
      $current_tag = '';
      break;
    case 'URL':
      $temp['url'] = $data;
      $current_tag = '';
      break;
    default:
      break;
  }
}

// de volgende functie schrijft de url en de titel uit van de
// huidige inhoud van de tijdelijke array...
function return_page() {
  global $temp;
  echo 'o <A HREF="'. $temp['url'] .'">'. $temp['title'] . '</A><BR>';
}

// het eigenlijke programma start hier...
$xml_file = 'slashdot.xml';
$type = 'UTF-8'; // dit is de char-set.

// hier maken we een nieuwe xml-parser aan.
// $xml_parser bevat nadien een link naar de parser
// die we in iedere functie kunnen gebruiken
$xml_parser = xml_parser_create($type);

```

```

// enkele opties zetten van de parser.
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_parser_set_option($xml_parser, XML_OPTION_TARGET_ENCODING, 'UTF-8');

// de volgende functie koppelt 2 functies aan de start en eind-tags
xml_set_element_handler($xml_parser, 'startElement','endElement');

// en we koppelen de data-functie aan de parser
xml_set_character_data_handler($xml_parser, 'characterData');

// we openen de file die we gaan parsen
if (!$fp = fopen($xml_file, 'r')) {
    die("Fout tijdens het openen van $xml_file!\n");
}

// ... en we gaan die per blok van 4k gaan lezen en parsen
while ($data = fread($fp, 4096)) {
    if (!$data = utf8_encode($data)) {
        echo 'ERROR'."\n";
    }
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf( "XML error: %s op lijn nr %d\n\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}

// en de parser hebben we niet meer nodig
xml_parser_free($xml_parser);
?>

</body>
</html>

```

---

#### Het gebruik van de XML-parser

We zien duidelijk dat er eerst 3 functies aangemaakt worden. De parameters zijn allemaal opgelegd door de XML-library. Telkens als de element-functies worden opgeroepen krijgen we de parser door, de naam van de tag en een array met extra opties (die we hier niet gebruiken).

De data wordt behandeld door een extra functie. Hier krijgen we enkel de parser en de data terug. Attributen kunnen niet worden ingebakken in data, wat vrij logisch is.

Het programma gaat dan de XML-parser opzetten en de correcte functies koppelen aan de parser.

Daarna wordt de file geopend en gaan we telkens blokken van 4kb gaan lezen en door de parser "duwen". De parser doet verder alle werk voor ons.

### 2.2.3 Opmerkingen

XML is een opkomende techniek. Het formaat is internationaal aanvaard en wordt meer en meer gebruikt. Verschillende zaken worden heden ten dage geconfigureerd via XML-files. Het is een kleine moeite om je eigen configuratie of data ook in XML-files op te slaan en zo een gemakkelijke interface aan te bieden voor anderen die je werk moeten gebruiken. Het updaten van prijslijsten kan bijvoorbeeld perfect vanuit een administratieprogramma wanneer er XML gedumpt wordt en de XML-file ge-upload wordt naar de webpagina.

Let er echter wel op dat veel programmeurs geen data tussen de tags plaatsen, maar alle data in de attributen steken van de openingstag.

## 2.3 PDF

PDF is een standaard bestandsformaat dat vooral in de windows-wereld gekend is als “Acrobat” documenten. PDF staat voor “Portable Document Format”. PDF wordt binnen de grafische industrie gebruikt als een standaard om ontwerpen en dergelijke door te geven aan anderen, of om documenten te publiceren.

PDF wordt ontwikkeld door Adobe, een van de marktleiders op het gebied van grafische software.

Onder PHP kunnen we de PDF-bibliotheek `pdflib` koppelen zodat we binnen PHP ook pdf-documenten kunnen maken en deze doorsturen naar de browser van de bezoeker. Denken we bijvoorbeeld maar aan prijslijsten die up-to-date zijn, een pagina in printer-vriendelijk formaat of een wegbeschrijving met plannetjes. Wanneer we een e-commerce site opzetten kunnen we de facturen en zo in pdf-formaat klaarmaken voor de klanten. Het toont zeker een stuk professioneler.

We zullen 1 voorbeeldje zien van het gebruik van PDF documenten binnen PHP. De werkwijze is identiek aan die van grafische bestanden. We moeten dus eerst een andere “content-type” gaan vermelden met een `header()`-commando. Dit zorgt ervoor dat de browser het juiste programma gaat opstarten om de PDF weer te geven (meestal acrobat reader op een windowsmachine).

```
header("Content-type: application/pdf");
header("Content-Disposition: inline; filename=foo.pdf");
```

Denk eraan dat er bij het gebruik van `header()` geen data mag gestuurd worden naar de browser voor het `header()`-commando.

De enige moeilijkheid die hier nog kan voorkomen is het feit dat we moeten meten met PDF-maten. Een stukje uit de manual van `pdflib`:

```
... The default coordinate system (or default user space in PDF lingo) has the origin in the
lower left corner of the page, and uses the DTP point as unit: 1 pt = 1 inch / 72 = 25,4 mm
/ 72 = 0,3528 mm
```

Het maken van een pdf-document demonstreren we aan de hand van een voorbeeldje:

---

```
<?php
header("Content-type: application/pdf");
header("Content-Disposition: inline; filename=foo.pdf");

// een nieuw pdf-bestand maken...
$pdf = PDF_new();
PDF_open_file($pdf, "test.pdf");
PDF_set_info($pdf, "Author", "Uwe Steinmann");
PDF_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
PDF_set_info($pdf, "Creator", "See Author");
PDF_set_info($pdf, "Subject", "Testing");

// een pagina toevoegen
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
PDF_set_font($pdf, "Times-Roman", 30, "host");
PDF_set_value($pdf, "textrendering", 1);
PDF_show_xy($pdf, "Times Roman outlined", 50, 750);

// grafische objecten toevoegen
PDF_moveto($pdf, 50, 740);
PDF_lineto($pdf, 330, 740);
PDF_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
?>
```

---

### Een PDF-bestand aanmaken

We onderscheiden duidelijk enkele delen. Eerst maken we een pdf-bestand aan, gaan dat dan vullen met data dat ofwel tekst is, ofwel tekeningen en uiteindelijk sluiten we het bestand en sturen we het door naar de browser van de gebruiker.

Denk er echter wel aan dat de meeste hosting-providers het niet voorzien om PDF-bestanden aan te maken met PHP.

## Hoofdstuk 3

# Voorbeeld van het gebruik van PHP

Door de grote populariteit van PHP zijn er veel projecten die PHP gebruiken, ofwel om een volledig project in te maken, ofwel om een deel van het project erin te implementeren. In het laatste geval gaat dat dan meestal over de interface naar de gebruikers toe. We gaan enkele voorbeelden aanhalen die PHP gebruiken als een oplossing voor een probleem.

Let erop dat we hier slechts een glimp gaan geven van wat er allemaal mogelijk is met PHP. Voor een goed overzicht kan je kijken op [www.php.net/projects.php](http://www.php.net/projects.php) of gewoon gaan zoeken op Internet.

### 3.1 Library's

Je kan PHP gaan uitbreiden met verschillende bibliotheken, maar je kan ook PHP-code gaan toevoegen aan je eigen projecten waardoor je heel wat meer PHP-functies en functionaliteit verkrijgt. Een heel goed voorbeeld hiervan is het “Horde”-project.

#### 3.1.1 Horde

The Horde Application Framework provides classes for dealing with preferences, compression, browser detection, connection tracking, MIME, and more.

Het Horde-project stamt uit de tijd van PHP3. Onder PHP3 was er totaal geen ondersteuning voor bijvoorbeeld sessies. Horde bracht hier een oplossing door sessies zelf te gaan implementeren op een zodanige wijze dat het altijd herbruikbaar was. Onze eigen login-module geeft een basisidee van het sessie-management van Horde. Er zitten nog veel andere extra's in de library's van horde. Let erop dat het volledige horde-project dus PHP-code is en geen gecompileerde library die in PHP gelinkt zit!

Horde is te vinden op [www.horde.org](http://www.horde.org).

#### 3.1.2 Andere

Er bestaan verder nog duizenden extra kleine toevoegingen die allemaal iets speciaals doen. Onze login-klasse zouden we ook openbaar kunnen maken als een “gemakkelijke en simpele login-controle adh van een MySQL database”. Twee voorbeelden zijn [www.phpclasses.com](http://www.phpclasses.com) en [www.phpbuilder.com](http://www.phpbuilder.com). Op de laatste site worden er ook enkele case-studies gemaakt met extra commentaar over een bepaald onderwerp.

### 3.2 Webmail applicaties

Onder webmail-applicaties verstaan we een PHP-frontend voor een mailbox zoals we die kennen van bijvoorbeeld Hotmail. Twee grote voorbeelden zijn Imp ([www.horde.org/imp](http://www.horde.org/imp)) en Jawmail ([jawmail.sourceforge.net](http://jawmail.sourceforge.net)). Verder zijn er nog andere projecten, maar die zijn allemaal minder krachtig dan deze twee.

#### 3.2.1 Imp

Imp is een heel populaire webmail-frontend die gebruik maakt van de horde-bibliotheek. Imp is een project dat reeds enkele jaren loopt en was een van de eerste die zoveel functionaliteit bood. Daar het

gebaseerd is op het Horde-project wordt de Imp-code mooi gescheiden van code om bijvoorbeeld in te loggen, of om via javascript verschillende zaken gedaan te krijgen (nieuwe vensters openen,...). Imp heeft echter 1 groot nadeel. De setup van Imp is heel ingewikkeld en imp is dus niet aan te raden voor beginnende gebruikers. Er is ondersteuning voor verschillende talen.

Gebruikers van Imp zijn bijvoorbeeld `webmail.rug.ac.be` en `mail.rug.ac.be`.

### 3.2.2 Jawmail

Jawmail is eerder een recent project maar is desondanks een heel krachtige webmail applicatie. De installatie ervan verloopt bijzonder vlot en het gebruik is eenvoudig. Verder kunnen er heel gemakkelijke extra plugins toegevoegd worden aan Jawmail, waardoor speciale toevoegingen voor enkele situaties heel gemakkelijk is. Er is ondersteuning voor verschillende talen en skins.

## 3.3 Discussieforums

Een discussieforum is een plaats waar verschillende gebruikers kunnen chatten over bepaalde onderwerpen. Iedere gebruiker kan ergens iets posten, er zijn beheerders die berichten kunnen schrappen,...

### 3.3.1 Phorum

Phorum is een oude rot onder de PHP-forums. Het is een van de eerste forums die zo uitgebreid was. Phorum is heel populair op sites die reeds langere tijd PHP gebruiken, aangezien het in die tijd de enige keuze was (wat niet wil zeggen dat het ingelijk niet goed is).

Er is ondersteuning voorzien voor verschillende talen en verschillende skins.

Phorum kan je vinden op [www.phorum.org](http://www.phorum.org).

### 3.3.2 phpBB

De website van phpBB vertelt hetvolgende over zichzelf:

phpBB is a high powered, fully scalable, and highly customizable forums package. phpBB has a user-friendly interface, simply and straight-forward administration panel, and helpfull FAQ. All this and being backed by the blazingly fast MySQL database server makes phpBB the ideal community solution for all web sites

phpBB beschikt inderdaad over een gemakkelijke administratie en is heel overzichtelijk geprogrammeerd. Er zijn enkele extra's ingevoegd zoals het gebruik van smiley's en speciale "bbCode" om extra's toe te voegen aan je berichten.

Verder zit er ook een goede zoekfunctie ingebakken.

phpBB staat voor "php bulletin board" en kan gevonden worden op [www.phpbb.com](http://www.phpbb.com).

## 3.4 Database administratie

Er bestaan enkele projecten die een database-administratie proberen op te zetten via PHP, maar enkel phpMyAdmin is hierin volledig geslaagd. Met phpMyAdmin kan je een MySQL-database volledig gaan beheren. De toegangsrechten hangen af van de username en het password dat je meegeeft met phpMyAdmin, dus kunnen we dit ook perfect gaan gebruiken om de gebruiker een gemakkelijke interface te geven naar hun database op een server.

phpMyAdmin kan gevonden worden op [www.phpwizard.net/projects/phpMyAdmin/](http://www.phpwizard.net/projects/phpMyAdmin/).

Er bestaan nog andere database-frontends, zoals phpPGAdmin, wat identiek is aan phpMyAdmin, maar dan voor PostgreSQL databases. Dit project kan gevonden worden op [www.greatbridge.org/project/phppgadmin/](http://www.greatbridge.org/project/phppgadmin/).

## 3.5 Content management systemen

Deze systemen zorgen ervoor dat je via een gemakkelijke interface de inhoud van je webpagina kan aanpassen. Er is bijna geen kennis van HTML vereist en iedere pagina ziet er identiek uit qua layout.

### 3.5.1 Basit

Basit, dat te vinden is op [www.fazlamesai.net/basit/eng/](http://www.fazlamesai.net/basit/eng/) biedt een eenvoudige interface om kleine sites gemakkelijk te onderhouden. Er is echter geen mogelijk om meerdere logins te gebruiken of om de artikels in te delen in verschillende delen.

### 3.5.2 Midgard

Midgard is 1 van de betere systemen om je website te onderhouden. Je kan er verschillende onderdelen van je site mee configureren, extra pagina's toevoegen, je site onderverdelen in verschillende stukken, ...

Midgard kan gevonden worden op [www.midgard-project.com](http://www.midgard-project.com).

### 3.5.3 PHPCMS

Dit project is ook veelbelovend. Momenteel is de site enkel maar in het Duits, maar de documentatie is volledig engelstalig. PHPCMS is een volledig site-administratie-tool. We kunnen er bladeren door de webpagina (filemanager), statistieken bekijken van de webpagina, maar natuurlijk ook files gaan toevoegen aan de website. PHPCMS is constant onder ontwikkeling.

[mcyra.homeip.net/homepage/phpcms/index.htm](http://mcyra.homeip.net/homepage/phpcms/index.htm) voor meer informatie.