12 Hoofdstuk

Html-sjablonen in PHP

Een sjabloon (in PHP template genaamd) bevat de volledige opmaak van een (html-)pagina. Code en opmaak zijn hierbij gescheiden. Terwijl veel startende programmeurs code en opmaak door elkaar gebruiken, gaan ervaren ontwikkelaars er meestal toe over deze te scheiden. Bij het ontwikkelen van grote, lastiger te onderhouden websites (of webapplicaties) wordt in de regel altijd gebruikgemaakt van templates. Dit komt het overzicht en de kwaliteit enorm ten goede. Er zijn veel verschillende templatesystemen voor PHP. Smarty is echter de bekendste en het is een officieel PHP-project.

U leert in dit hoofdstuk:

In dit hoofdstuk gaan we dieper in op Smarty-templates. U leert het principe van templates kennen. We zullen de belangrijkste eigenschappen van Smarty bespreken. Aan de hand van voorbeelden leert u hoe het systeem werkt en hoe u het zelf kunt gebruiken.

Waarom templates?

Er zijn verschillende redenen waarom we gebruikmaken van templates. De belangrijkste zijn:

- Code en opmaak worden gescheiden.
- Opmaak kan onafhankelijk van code worden gemaakt (vormgever).
- Opmaak kan onafhankelijk van code worden aangepast.
- Er kunnen verschillende templates aan één systeem worden gekoppeld (voor meer sites).

Meer informatie

Smarty is een officieel PHP-project. U vindt het project op **smarty.php.net**.

Smarty installeren

Uitpakken

Smarty is ingepakt met de tar.gz-methode. U kunt deze bestanden bijvoorbeeld uitpakken met het opensourceprogramma 7-Zip. U vindt het op **www.7-zip.org**.

Smarty laat zich eenvoudig installeren. U downloadt het bestand van **smarty.php.net/download.php**. Voor de installatie gaat u als volgt te werk:

1. Pak het bestand uit.
2. De directory waar het om gaat heet libs. Kopieer deze directory naar uw webroot (bijvoorbeeld /vhost/www.leer-php.nl/www).
3. Maak een directory met de naam templates.
4. Maak een directory met de naam templates_c.
5. Maak een directory met de naam configs.
6. Maak een directory met de naam cache.
7. Maak de directory's templates_c en cache eigendom van de webserver (bijvoorbeeld user nobody of apache).
8. Zet schrijfrechten op de directory's templates_c en cache (rechtencode 775).

Beginnen met Smarty

Zoals gezegd scheiden we code en opmaak. Dat betekent dat we in de code bepalen waaruit de inhoud (niet de opmaak) van de template bestaat. We halen bijvoorbeeld de juiste inhoud uit de database en zetten die klaar voor de template. In de template verwijzen we vervolgens weer met een bepaalde sleutel naar de inhoud. Laten we het templateprincipe eens bekijken aan de hand van een eenvoudig voorbeeld.

templ1.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Maak object
$smarty = new Smarty;
// We kennen de variabelen toe
$smarty->assign('naam', 'Leer jezelf professioneel PHP');
$smarty->assign('prijs', '24.90');
// display it
$smarty->display('template1.tpl');
?>
```

Paden instellen in Windows

In Windows ziet het pad er net iets anders uit dan onder Linux. De verwijzing naar ./libs ziet er dan als volgt uit:
`ini_set('include_path', '.;.\libs'.$pad);`

Het Smarty-systeem is in feite een extern PHP-bestand dat we eerst moeten includen. In het voorbeeld gaan we ervan uit dat alle Smarty-directory's (libs, templates, configs, templates_c en cache) in dezelfde directory staan als ons script. Omdat we includen uit verschillende directory's, passen we eerst het includepad aan. We lezen hiervoor eerst het huidige pad in en voegen ./libs daaraan toe. Vervolgens includen we de Smarty-klasse.

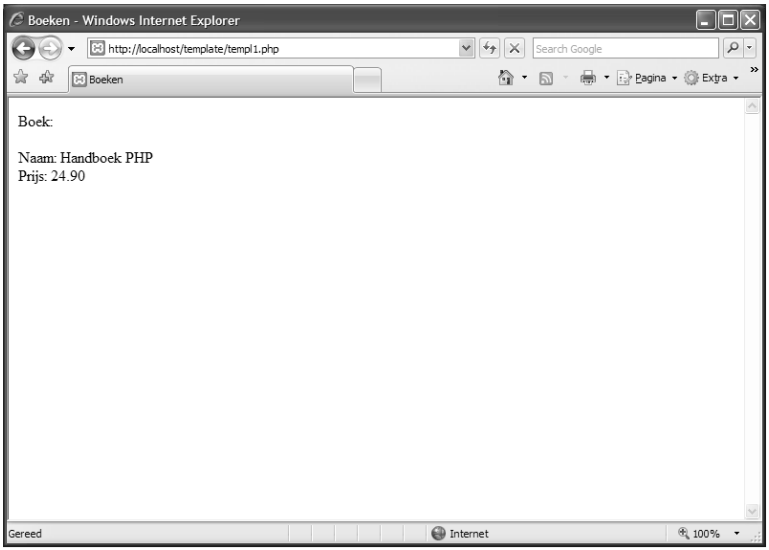
Smarty gedraagt zich als een klasse. Voor ons betekent dit dat we eerst een object moeten instantiëren met $smarty = new Smarty. Vervolgens kunnen we verschillende functies van dit object benaderen via $smarty, dus: $smarty->display('template1.tpl'). Het is niet zo belangrijk om te weten hoe

zo'n object precies werkt. Het is vooral belangrijk te weten dat je het eerst moet aanmaken (instantiëren) en hoe de methoden van het object gebruikt kunnen worden.

Met de methode assign van $smarty kennen we een variabele toe aan onze template. Door middel van de methode display geven we aan dat we alle toegekende elementen willen toepassen op template template1.tpl.

De template template1.tpl slaan we op in de directory templates. Het bestand ziet er als volgt uit:

```
<html>
<head>
<title>Boeken</title>
</head>
<body>
Boek:<p>
Naam: {$naam}<br>
Prijs: {$prijs}<br>
</body>
</html>
```



Afbeelding 12.1 *Een eerste Smarty-voorbeeld.*

In eerste instantie lijkt het alsof we hier te maken hebben met een gewoon html-document. In zekere zin klopt dat ook, maar toch vallen twee dingen op. De waarden tussen accolades verwijzen naar de variabelen die we gedefinieerd hebben in het voorgaande PHP-script. Het templatesysteem vervangt {$naam} dus door de waarde die we in het PHP-script toegekend hebben. Voor {$prijs} geldt hetzelfde. In het algemeen gelden de volgende regels voor specifieke Smarty-tags:

- Smarty-tags staan tussen { en }. Deze tags zijn bij elke Smarty-handeling noodzakelijk.
- Een variabele wordt aangeduid met $variabele.

Wanneer u template1.php uitvoert, dan zult u het gewenste resultaat op het scherm zien: achter Naam staat keurig de waarde die we in het script hebben gedefinieerd, en voor Prijs geldt hetzelfde.

Alternatieve bestandslocaties

In dit boek gaan we ervan uit dat de Smarty-directory's in dezelfde directory staan als de scripts. Wanneer u een andere directory-indeling wenst, kunt u aangeven dat u andere locaties wilt gebruiken. U doet dit op de volgende manier:

```
<?php
// volledige pad naar Smarty
require('/usr/local/lib/php/Smarty/Smarty.class.php');
$smarty = new Smarty();
$smarty->template_dir = '/web/www.domain.nl/smarty/templates';
$smarty->compile_dir = '/web/www.domain.nl/smarty/templates_c';
$smarty->cache_dir = '/web/www.domain.nl/smarty/cache';
$smarty->config_dir = '/web/www.domain.nl/smarty/configs';
$smarty->assign('variabele', 'Waarde');
$smarty->display('index.tpl');
?>
```

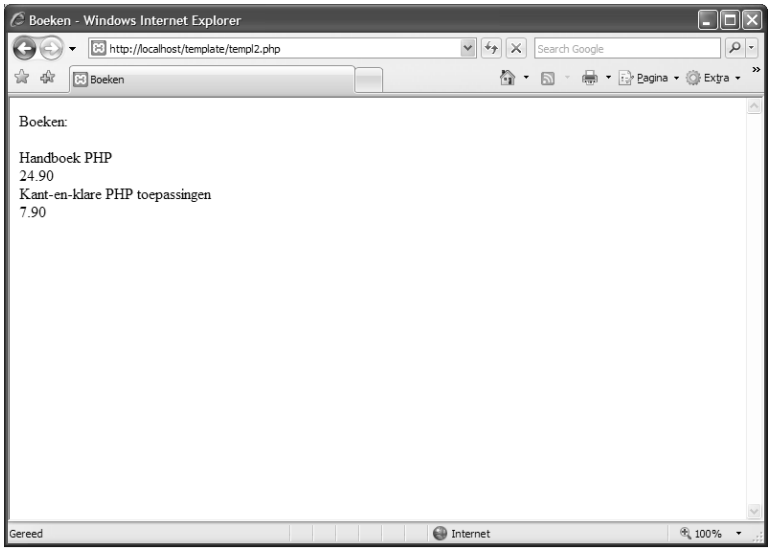
Meerdimensionale variabele doorgeven

In het eerste voorbeeld hebben we eendimensionale variabelen doorgegeven. Natuurlijk willen we in de praktijk vaak lijsten (afkomstig uit een database) doorgeven aan het templatesysteem. We gebruiken hiervoor arrays. In het volgende script gaan we uit van een lijst met boeken. Elk boek bevat in ieder geval een naam en een prijs. We definiëren elk boek apart als array. Alle boeken samen vormen ook een array. Eén afzonderlijk boek is dus een array in een array. Als script ziet dit er als volgt uit:

templ2.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', array(
array('naam'=>'Handboek PHP',
                    'prijs'=>'24.90'),
                array('naam'=>'Kant-en-klare PHP-toepassingen',
                    'prijs'=>'7.90')));
// display it
$smarty->display('template2.tpl');
?>
```

In de bijbehorende template maken we gebruik van een sectie. Binnen deze sectie doorlopen we de boekenarray. Omdat we binnen een sectie verwijzen naar een bepaald element in de boekenarray, moeten we gebruikmaken van de indexmogelijkheden van Smarty. De tag {$boeken[mijnsectie]. naam} wil zeggen: verwijs naar de naam van de huidige index binnen de sectie mijnsectie van de variabele boeken. De eerste keer dat de sectie doorlopen wordt bevat {$boeken[mijnsectie].naam} dus de waarde 'Leer jezelf professioneel PHP', de tweede keer 'Kant-en-klare PHP-toepassingen'.



Afbeelding 12.2 *Doorloop een sectie met array $boeken.*

De template ziet er als volgt uit:

template2.tpl

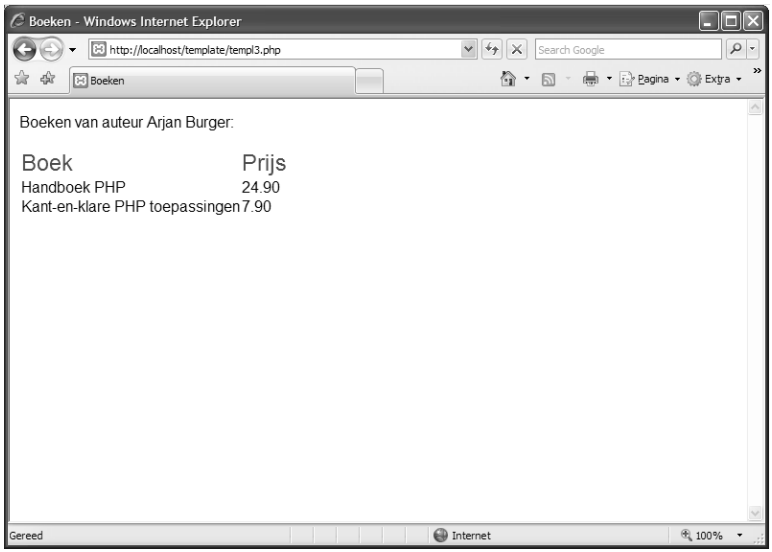
```
<html>
<head>
<title>Boeken</title>
</head>
<body>
Boeken:<p>
{section name=mijnsectie loop=$boeken}
    {$boeken[mijnsectie].naam}<br />
    {$boeken[mijnsectie].prijs}<br />
{/section}
</body>
</html>
```

Opmaak van een template

Tot nu toe hebben we de opmaak van de template erg eenvoudig gehouden. Natuurlijk kunnen we alle mogelijkheden gebruiken die html biedt. Bekijk het volgende script en de template eens.

templ3.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', array( array( 'naam'=>'Handboek PHP',
                  'prijs'=>'24.90'),
              array('naam'=>'Kant-en-klare PHP-toepassingen',
                  'prijs'=>'7.90')));
$smarty->assign('auteur', 'arjan burger');
// display it
$smarty->display('template3.tpl');
?>
```



Afbeelding 12.3 *Een echt html-template.*

Dit script lijkt erg op het vorige script, behalve dat we ook de variabele auteur meesturen. De naam is bewust in kleine letters gezet. In de template zullen we namelijk zien dat Smarty ook functies bevat die direct aan variabelen gekoppeld zijn. De tag {$auteur|capitalize} geeft namelijk aan dat de variabele $auteur afgedrukt moet worden, maar dat deze tevens van hoofdletters voorzien moet worden. Dit wil zeggen dat alle eerste letters van alle woorden binnen de variabelen naar een hoofdletter worden omgezet. Dit is vergelijkbaar met de functie ucwords in PHP.

template3.tpl

```
<html>
<head>
<title>Boeken</title>
{literal}
<style type="text/css">
body, td {font-family: Arial }
td.kop {font-size: 18pt; color: red}
</style>
{/literal}
</head>
<body>
Boeken van auteur {$auteur|capitalize}:<p>
<table border="0" cellpadding="0">
<tr>
   <td class="kop">Boek</td>
   <td class="kop">Prijs</td>
</tr>
{section name=mijnsectie loop=$boeken}
<tr>
   <td>{$boeken[mijnsectie].naam}</td>
     <td>{$boeken[mijnsectie].prijs}</td>
</tr>
{/section}
</table>
</body>
</html>
```

{literal}

Wanneer we niet willen dat een bepaald gedeelte van de template door Smarty wordt geïnterpreteerd, dan gebruiken we de tag {literal}. Alles tussen {literal} en {/literal} wordt overgeslagen. Dit is praktisch wanneer we bijvoorbeeld tekens gebruiken die in Smarty ook worden gebruikt (zoals {tag}).

Gegevens verwerken in een template

We hebben gezien dat we data in variabelen doorgeven aan de template. Wanneer data afkomstig is uit een database hebben we bijna altijd te maken met een meerdimensionale variabele. Het principe werkt eigenlijk als volgt: eerst halen we de data uit de database en vervolgens brengen we deze onder in een array. Het meest praktisch is om deze veel voorkomende handeling te automatiseren. Daarom treft u hierna een script aan dat een willekeurige query uitvoert, de resultaten ophaalt en verwerkt in een meerdimensionale, associatieve array.

```
<?php
function data2array($sql) {
    $resultaat = mysql_query($sql) or die("Kan de query niet uitvoeren!");
    // zet de velden bovenaan de tabel
    $aantal = mysql_num_fields($resultaat);
    for ($nr=0; $nr < $aantal; $nr++) {
        $veldnaam[$nr] = strtolower(mysql_field_name($resultaat, $nr));
    }
    $rijen = mysql_num_rows($resultaat);
    $rijen_resultaat = array();
    if ($rijen > 0) {
        for ($nr=0; $nr < $rijen; $nr++) {
            $deze_rij = array();
            for ($velden=0; $velden < $aantal; $velden++) {
                $waarde = mysql_result($resultaat, $nr, $velden);
                $deze_rij[$veldnaam[$velden]] = $waarde;
            }
            $rijen_resultaat[] = $deze_rij;
        }
    }
    return $rijen_resultaat;
}
?>
```

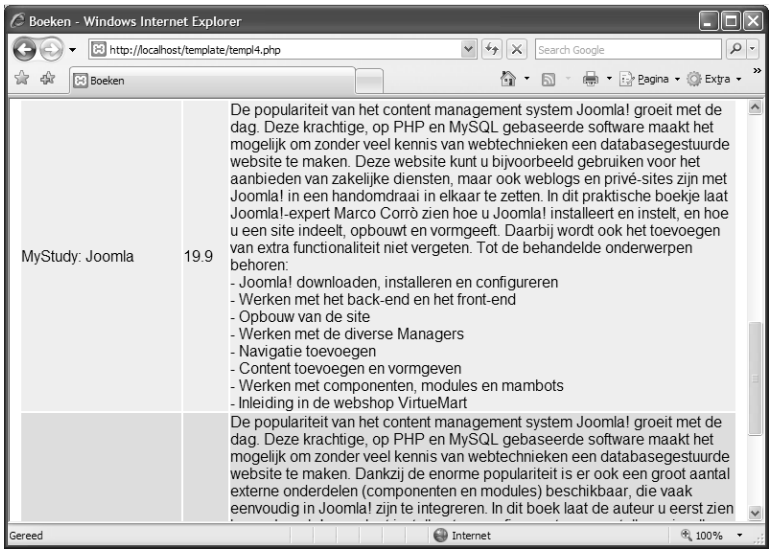
De functie data2array bevat in principe geen nieuwe zaken. Deze leest het resultaat van de query in, haalt vervolgens de velden op en leest dan de rijen in. Dan wordt een verbinding gemaakt tussen veld en waarde. Elke rij wordt apart opgeslagen in de array $deze_rij. Wanneer de hele rij is samengesteld, wordt deze toegevoegd aan de array $rijen_resultaat. Deze array geldt tevens als resultaat van de functie.

Laten we de functie data2array eens uitproberen in de praktijk. In het volgende script lezen we alle velden in van de tabel Artikel. Vervolgens sturen we deze als de variabele boeken door naar de template.

templ4.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
include('data2array.php');
//maak connectie
$db = mysql_connect("localhost", "leerphp", "geheim")
        or die("Kan niet verbinden: " . mysql_error());
mysql_select_db("leerphp", $db);
//definieer query voor alle boeken
$sql = "SELECT *
FROM Artikel
";
$boekenarray = data2array($sql);
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', $boekenarray);
$smarty->assign('auteur', 'arjan burger');
// display it
$smarty->display('template4.tpl');
?>
```

In de bijbehorende template gebruiken we de functie section om de array af te drukken. Nieuw is de functie cycle. Deze functie doorloopt een reeks waarden. Elke keer wanneer een lus (zoals section) wordt uitgevoerd, kiest cycle de volgende waarde (of de eerste, wanneer de reeks doorlopen is). In ons geval wordt de cycle-tag de eerste keer dus vervangen door #eeeeee, de tweede keer door #dddddd en dan weer door #eeeeee enzovoort.



Afbeelding 12.4 *Een selectie uit de database.*

template4.tpl

```
<html>
<head>
<title>Boeken</title>
{literal}
<style type="text/css">
body, td {font-family: Arial }
td.kop {font-size: 18pt; color: red}
</style>
{/literal}
</head>
<body>
Boeken van auteur {$auteur|capitalize}:<p>
<table border="0" cellpadding="0">
<tr>
   <td class="kop">Boek</td>
   <td class="kop">Prijs</td>
   <td class="kop">Omschrijving</td>
</tr>
{section name=mijnsectie loop=$boeken}
<tr bgcolor="{cycle values="#eeeeee,#dddddd"}">
   <td>{$boeken[mijnsectie].naam}</td>
     <td>{$boeken[mijnsectie].prijs}</td>
```

```
    <td>{$boeken[mijnsectie].omschrijving}</td>
</tr>
{/section}
</table>
</body>
</html>
```

Caching

Smarty kan gebruikmaken van caching. Dat wil zeggen dat het systeem niet elke keer opnieuw een pagina genereert, maar gebruik kan maken van een reeds gegenereerde pagina. Dit kan – zeker bij grotere sites – behoorlijke snelheidswinsten opleveren.

Nut van caching

Caching heeft alleen zin wanneer veel gebruikers dezelfde pagina's te zien krijgen. Anders levert het geen snelheidswinst (processortijd) op.

In het volgende script (dezelfde template als hiervoor) maken we gebruik van caching. We geven dit op de volgende manier door aan Smarty:

```
$smarty->caching = true;
```

Door middel van een if-constructie kunnen we kijken of er een cacheversie van ons template aanwezig is. Smarty kent hiervoor de methode $smarty->is_cached($template). Wanneer de template al in de cache staat, hoeven we de parameters die nodig zijn om de pagina te genereren namelijk niet meer aan te maken.

templ5.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
include('data2array.php');
//maak connectie
$db = mysql_connect("localhost", "leerphp", "geheim")
        or die("Kan niet verbinden: " . mysql_error());
```

```
mysql_select_db("leerphp", $db);
//definieer query voor alle boeken
$sql = "SELECT *
FROM Artikel
";
$boekenarray = data2array($sql);
// Maak object
$smarty = new Smarty;
$smarty->caching = true;
// see if the page is already cached
if(!$smarty->is_cached('template4.tpl')) {
    // Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
    $smarty->assign('boeken', $boekenarray);
    $smarty->assign('auteur', 'arjan burger');
}
// display it
$smarty->display('template4.tpl');
?>
```

Levensduur

We kunnen per cachepagina bepalen hoe lang deze mag bestaan. Wanneer template4.tpl bijvoorbeeld maximaal vijf minuten mag bestaan, dan geven we dit aan Smarty door met $smarty->cache_lifetime = 5*60; (tijd in seconden).

Templates koppelen

Ook binnen templates kunnen we gebruikmaken van includebestanden. Zo kunnen we bijvoorbeeld gemakkelijk een header en footer gebruiken. In het volgende script roepen we template6.tpl aan. Dit template maakt gebruik van een header en een footer.

templ6.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
include('data2array.php');
```

```
//maak connectie
$db = mysql_connect("localhost", "leerphp", "geheim")
        or die("Kan niet verbinden: " . mysql_error());
mysql_select_db("leerphp", $db);
//definieer query voor alle boeken
$sql = "SELECT *
FROM Artikel
";
$boekenarray = data2array($sql);
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', $boekenarray);
$smarty->assign('auteur', 'arjan burger');
// display it
$smarty->display('template6.tpl');
?>
```

De basistemplate ziet er als volgt uit:

template6.tpl

```
{include file="header.tpl" title="Boekenlijst"}
Boeken van auteur {$auteur|capitalize}:<p>
<table border="0" cellpadding="0">
<tr>
    <td class="kop">Boek</td>
    <td class="kop">Prijs</td>
    <td class="kop">Omschrijving</td>
</tr>
{section name=mijnsectie loop=$boeken}
<tr bgcolor="{cycle values="#eeeeee, #dddddd"}">
    <td>{$boeken[mijnsectie].naam}</td>
        <td>{$boeken[mijnsectie].prijs}</td>
    <td>{$boeken[mijnsectie].omschrijving}</td>
</tr>
{/section}
</table>
{include file="footer.tpl"}
```

Met behulp van de functie include kunnen we opgeven welk bestand ingelezen moet worden. Bij het aanroepen van header.tpl wordt ook een variabele title meegegeven. Deze variabele kunnen we benaderen in header.tpl. Deze wordt in dit geval gebruikt om de titel van de pagina te definiëren.

header.tpl

```
<head>
<title>{$title|default:"Test"}</title>
{literal}
<style type="text/css">
body, td {font-family: Arial }
td.kop {font-size: 18pt; color: red}
</style>
{/literal}
</head>
<body>
```

footer.tpl

```
</body>
</html>
```

Gebruik van $title

In header.tpl drukken we de variabele $title (afkomstig uit template6.tpl) af. Wanneer geen variabele title is meegegeven, maken we gebruik van de standaardwaarde Test. De constructie ziet er uit als {$variabele|default:"Standaardwaarde"}.

Constanten in configuratiebestanden

In PHP kennen we het begrip constante. Dit is een begrip dat we ergens definiëren met een vaste waarde, zoals een pad naar een bestand, een kleur, enzovoort. Binnen Smarty kunnen we constanten vastleggen in een configuratiebestand. Het volgende script roept een template aan dat gebruikmaakt van constanten.

templ7.php

```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
include('data2array.php');
```

```
//maak connectie
$db = mysql_connect("localhost", "leerphp", "geheim")
        or die("Kan niet verbinden: " . mysql_error());
mysql_select_db("leerphp", $db);
//definieer query voor alle boeken
$sql = "SELECT *
FROM Artikel
";
$boekenarray = data2array($sql);
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', $boekenarray);
// display it
$smarty->display('template7.tpl');
?>
```

Constanten in een template

Constanten geeft u in de template weer met {#constante#}.

Gebruikte directory

De configuratiebestanden (zoals basisconfig.conf) staan standaard in de directory configs.

In de volgende template importeren we door middel van de functie config_load de constanten uit het bestand basisconfig.conf.

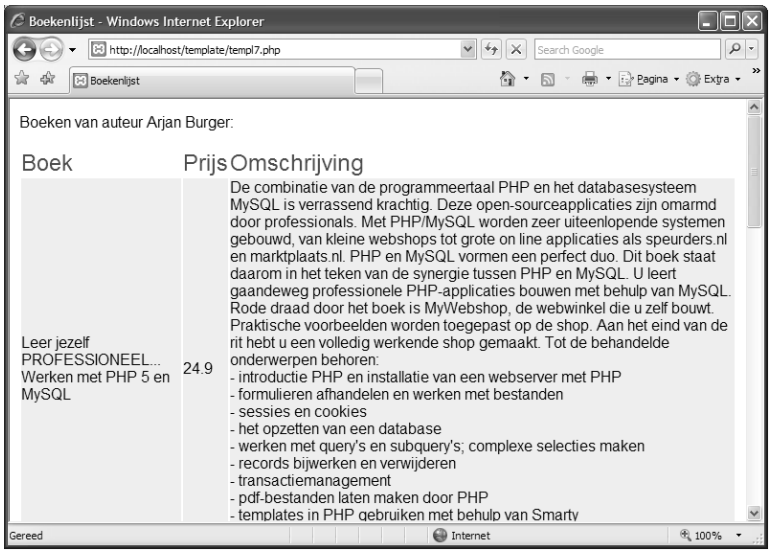
template7.tpl

```
{config_load file="basisconfig.conf"}
{include file="header.tpl" title="Boekenlijst"}
Boeken van auteur {#auteur#}:<p>
<table border="0" cellpadding="0">
<tr>
    <td class="kop">Boek</td>
    <td class="kop">Prijs</td>
    <td class="kop">Omschrijving</td>
</tr>
{section name=mijnsectie loop=$boeken}
<tr bgcolor="{cycle values="#eeeeee,#dddddd"}">
    <td>{$boeken[mijnsectie].naam}</td>
        <td>{$boeken[mijnsectie].prijs}</td>
```

```
    <td>{$boeken[mijnsectie].omschrijving}</td>
</tr>
{/section}
</table>
Als u nu bestelt, dan krijgt u {#korting#}% korting!
{include file="footer.tpl"}
```

basisconfig.conf

```
# basisconfiguratie
# definieer hier de standaardwaarden
auteur = "Arjan Burger"
korting = 10
```



Afbeelding 12.5 *Gebruik van constanten in templates.*

Condities

In principe is een template bedoeld om zo weinig mogelijk te programmeren. Er bestaan echter wel degelijk mogelijkheden om bijvoorbeeld if…else-constructies te gebruiken. Zo'n constructie ziet er in Smarty als volgt uit:

```
{if conditie}
    …template…
{else}
    …template…
{/if}
```

Een conditie heeft vrijwel dezelfde vorm als in PHP. Een paar voorbeelden:

```
{if $boeken[mijnsectie].aantal >= 5}
{if $naam == "Arjan"}
{if $waarde < 3}
```

In het volgende script bekijken we welk aantal een individuele klant van een artikel heeft besteld. Wanneer het aantal groter is dan vijf, wordt het in een afwijkende kleur weergegeven.

templ8.php

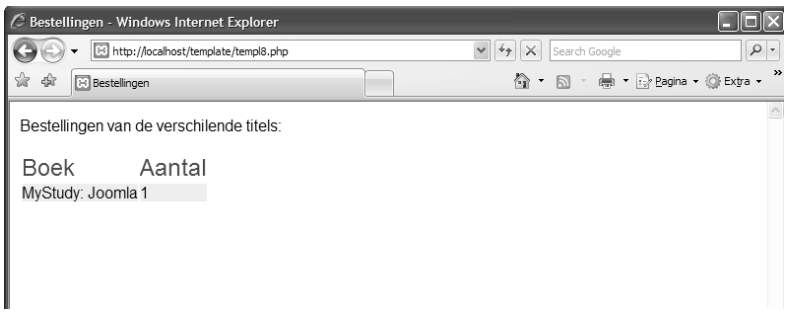
```
<?php
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
include('data2array.php');
//maak connectie
$db = mysql_connect("localhost", "leerphp", "geheim")
        or die("Kan niet verbinden: " . mysql_error());
mysql_select_db("leerphp", $db);
//definieer query voor alle boeken
$sql = "SELECT Naam, Aantal
FROM Artikel, Bestelling
WHERE Artikel.artikel_id = Bestelling.Artikel_id
GROUP BY Aantal
";
$boekenarray = data2array($sql);
```

```
// Maak object
$smarty = new Smarty;
// Ken verschillende boeken en prijzen toe aan de smarty variabele boeken
$smarty->assign('boeken', $boekenarray);
// display it
$smarty->display('template8.tpl');
?>
```

De template ziet er als volgt uit:

template8.tpl

```
{include file="header.tpl" title="Bestellingen"}
Bestellingen van de verschilende titels:<p>
<table border="0" cellpadding="0">
<tr>
   <td class="kop">Boek</td>
   <td class="kop">Aantal</td>
</tr>
{section name=mijnsectie loop=$boeken}
<tr bgcolor="{cycle values="#eeeeee, #dddddd"}">
   <td>{$boeken[mijnsectie].naam}</td>
   {if $boeken[mijnsectie].aantal >= 5}
          <td><font color="red">{$boeken[mijnsectie].aantal}</font></td>
   {else}
          <td>{$boeken[mijnsectie].aantal}</td>
   {/if}
</tr>
{/section}
</table>
{include file="footer.tpl"}
```



Afbeelding 12.6 *Alleen aantallen groter dan vijf worden in een afwijkende kleur weergegeven.*

Door de if-constructie bekijken we of van een artikel vijf of meer exemplaren zijn besteld. Is dat het geval dan wordt de code tussen {if} en {else} uitgevoerd. Is dat niet het geval, dan wordt de code tussen {else} en {/if} uitgevoerd.

MyWebshop met templates

MyWebshop gebruikt standaard geen templates. Om te laten zien dat templates ook hier meerwaarde kunnen hebben, bouwen we een aantal bestanden om. We starten met de index. Deze pagina laat een bovenbalk zien met een aanbieding. Deze aanbieding kan variëren. In de oude situatie werd willekeurig één van de drie aanbiedingsbestanden naar het scherm gestuurd. Wanneer we toch met templates werken, is het handiger om de teksten van de aanbiedingen in het script te houden en het tonen ervan in een template uit te voeren.

Smarty-includes

Ook de Smarty-includes zijn opgenomen in config.php, omdat deze vrijwel in elk script gebruikt worden.

Switch

In index.php wordt door middel van een switch-constructie de aanbieding in Smarty-variabelen vormgegeven.

Alle configuratiefuncties van de webshop hebben we nu in een apart configuratiebestand gezet, omdat de functie van header.php als html-bestand niet meer bestaat. Het configuratiebestand ziet er als volgt uit:

config.php

```
<?php
session_start();
require "../../mysqldb.php";
$sitepad = "/uploadimages/"; // voor afbeeldingen in site
$pad = "/vhost/prof.leer-php.nl/home/www/html/uploadimages/"; // plaats afbeeldingen op server
// Zet een pad naar Smarty (./libs)
$pad = ini_get('include_path');
ini_set('include_path', './libs:'.$pad);
// Haal Smarty binnen
include('Smarty.class.php');
// Haal data2array functie binnen
```

```
include('data2array.php');
?>
index.php
<?php
include "config.php";
// Maak object
$smarty = new Smarty;
$nummer = mt_rand(0,2); // bepaalt een random getal tussen 0 en 2
// selecteer de aanbieding
switch($nummer) {
    case 0:
        $smarty->assign('kop', 'CD aanbieding!');
        $smarty->assign('melding', 'Deze week is er een bijzondere CD in de aanbieding.
            Bestel deze meteen!');
        break;
    case 1:
        $smarty->assign('kop', 'Shampoo aanbieding!');
        $smarty->assign('melding', 'Deze week is de shampoo in de aanbieding.
            Bestel deze meteen!');
        break;
    case 2:
        $smarty->assign('kop', 'Weekaanbieding!');
        $smarty->assign('melding', 'Wij hebben deze week een bijzondere weekaanbieding voor u!');
        break;
}
// display it
$smarty->display('index.tpl');
?>
```

index.tpl

```
{include file="header.tpl"}
<div class="kop">
{$kop}
</div>
{$melding}
{include file="footer.tpl"}
```

Uiteraard maken alle webshoptemplates gebruik van een header (header.tpl) en een footer (footer.tpl). Hierin worden onder meer de knoppenbalk en de stylesheet meegenomen.

header.tpl

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> MyWebshop </TITLE>
{literal}
<script language="javascript">
function Confirm(i)
{
    var conmessage = new Array(3)
    conmessage[0] = "Weet u zeker dat u de bestelling wilt uitvoeren?"
    conmessage[1] = "Weet u zeker dat u dit artikel wilt wissen?"
    conmessage[2] = "Wilt u deze bestelling verwijderen?"
    var DoConfirm = confirm(conmessage[i]);
    if (DoConfirm)
    return true ;
    else
    return false ;
}
</script>
<style type="text/css">
body, td, form, input, submit, select {font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 12px; color: black}
a {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #333333}
.big {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 16px; color: #EE153C}
.kop {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12px; color: #EE153C}
.main {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 16px; color: white}
.mainkop {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #F1F3D1}
</style>
{/literal}
</HEAD>
<BODY topmargin=0 leftmargin=0 marginheight=0 marginwidth=0>
<table border=0 width=100% cellpadding=0 cellspacing=0>
    <tr>
        <td width=2% bgcolor="#1104FF"> </td>
        <td colspan=2 bgcolor="#1104FF" class="main">MyWebshop
            <a href="zoek.php" class="mainkop">[zoek]</a> <a href="winkelwagen.php"
             class="mainkop">[winkelwagen]</a> <a href="admin.php"
             class="mainkop">[admin]</a> <a href="logout.php"
             class="mainkop">[logout]</a></td>
    </tr>
    <tr>
        <td colspan=2> </td>
```

```
    </tr>
    <tr>
        <td width=2%> </td>
        <td height=100% width=90%>
```

footer.tpl

```
        </td>
        </tr>
</table>
</BODY>
</HTML>
```

Literal

De JavaScript- en stylesheetgedeelten in de header zijn gedefinieerd als literal, omdat dit niet door Smarty geparsed hoeft te worden.

Afbeelding 12.7 *Index MyWebshop in templateversie.*

De zoekfunctie

De zoekfunctie is redelijk eenvoudig te maken. Zoek.php bestaat voornamelijk uit een query die de categorieën uit de shop toont. We kunnen deze query omzetten naar een array door middel van de functie data2array. Het script ziet er als volgt uit:

zoek.php

```
<?php
require "config.php";
$sql = "SELECT * FROM Categorie";
$catsarray = data2array($sql);
```

```
// Maak object
$smarty = new Smarty;
// Ken verschillende categorieen toe aan variabele cats
$smarty->assign('cats', $catsarray);
// display it
$smarty->display('zoek.tpl');
?>
```

Afbeelding 12.8 *De zoekfunctie in templates.*

De bijbehorende template zoek.tpl laat de resultaten van de query zien en bouwt url's op die verwijzen naar show.php. In show.php worden de artikelen uit de geselecteerde categorie vervolgens getoond.

zoek.tpl

```
{include file="header.tpl"}
Wij bieden de volgende categorie&euml;n artikelen aan:<br><p>
{section name=mijnsectie loop=$cats}
<a href="show.php?catid={$cats[mijnsectie].categorie_id}&catnaam={$cats[mijnsectie].naam}"
   >{$cats[mijnsectie].naam}</a><br />
{/section}
<form action="show.php" method="post">
Zoek op artikel: <input type="text" name="zoek">
<input type="submit" name="knop" value="zoek!"> 
<input type="submit" name="knop" value="zoek geavanceerd!">
</form>
{include file="footer.tpl"}
```

Toon categorie met afbeeldingen

Het laatste script dat we omzetten is het script show.php. Dit script toont de inhoud van een bepaalde categorie met bijbehorende afbeeldingen. Nu kan één artikel meer afbeeldingen maken. Dat betekent dat het attribuut afbeelding van een artikel niet een waarde, maar een array is. Het volgende script leest alle artikelen uit een categorie in en plaatst ze in een array. Wanneer bij een bepaald artikel ook afbeeldingen horen, plaatst het script deze samen ook in een aparte array. Deze afbeeldingsarray wordt weer toegevoegd aan de oorspronkelijke artikelarray.

show.php

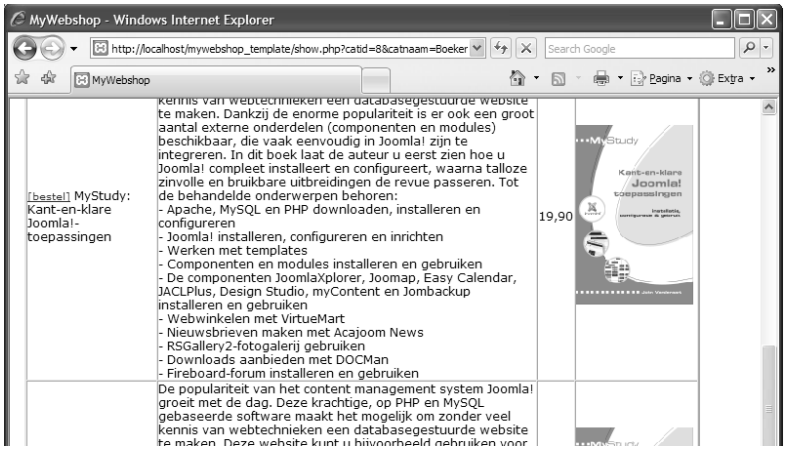
```
<?php
require "config.php";
// Maak object
$smarty = new Smarty;
$sql = "SELECT * FROM Categorie, Categorie_per_Artikel, Artikel
    WHERE Artikel.ARTIKEL_ID=Categorie_per_Artikel.ARTIKEL_ID
    AND Categorie.CATEGORIE_ID=Categorie_per_Artikel.CATEGORIE_ID
    AND Categorie_per_Artikel.CATEGORIE_ID=".$_GET["catid"];
$zoek = $_POST["zoek"];
if ($zoek && $_POST["knop"] == "zoek!") {
    // eenvoudig zoeken
    $sql = "SELECT * FROM Artikel WHERE Naam LIKE '%$zoek%'";
} elseif ($zoek && $_POST["knop"] == "zoek geavanceerd!") {
    // geavanceerd zoeken
    $sql = "SELECT *
            FROM Artikel
            WHERE MATCH (
            Naam, Omschrijving
            )
            AGAINST (
            '$zoek'
            IN BOOLEAN
            MODE
            )";
}
$resultaat  = mysql_query($sql); // voer SQL code uit
if (mysql_num_rows($resultaat) > 0) {
    $artikelen = array();
    $images = array();
    if ($zoek) {
        $smarty->assign('gezocht', "Gezocht op <b>$zoek</b><br><p>");
```

```
    } else {
        $smarty->assign('gezocht', "Categorie: ".$_GET["catnaam"]."<br><p>");
    }
    while ($rij = mysql_fetch_array($resultaat)) {
        $artikel = array();
        $images = array();
        $artikel["artikelid"] = $rij["ARTIKEL_ID"];
        $artikel["naam"]      = $rij["Naam"];
        $artikel["omschrijving"]= $rij["Omschrijving"];
        $artikel["prijs"]     = number_format($rij["Prijs"], 2, ',', ' ');
        $sql = "SELECT * FROM Afbeelding WHERE ARTIKEL_ID=".$rij["ARTIKEL_ID"];
            $afbeeldingen = mysql_query($sql);
            if (mysql_num_rows($afbeeldingen) > 0) {
                while ($afb_rij = mysql_fetch_array($afbeeldingen)) {
                        $bestandsnaam = $sitepad.$afb_rij["AFBEELDING_ID"].$afb_rij["Bestandstype"];
                        $images[] = $bestandsnaam;
                }
            $artikel["images"] = $images;
            }
        $artikelen[] = $artikel;
    }
} else {
    $smarty->assign('gezocht', "Er zijn geen artikelen die aan deze criteria voldoen");
}
// Voeg de verschillende artikelen toe
$smarty->assign('artikel', $artikelen);
// display it
$smarty->display('show.tpl');
?>
```

In de bijbehorende template introduceren we een nieuwe structuur: for...each. Deze structuur is ook bekend vanuit PHP. We kunnen er een array mee doorlopen en steeds de sleutel/waardeparen (key/item) gebruiken om bijvoorbeeld af te drukken. In het volgende voorbeeld doorlopen we met het statement section de array $artikel. Per artikel constateren we of er afbeeldingen aanwezig zijn (we checken of ($artikel[mijnsectie].images) een array is). Is dit het geval dan doorlopen we met for…each de array images die bij dat specifieke artikel behoort ($artikel[mijnsectie].images). De itemwaarden (uit de sleutel/waardeparen) gebruiken we om de url in een imagetag te plaatsen. Zijn er geen afbeeldingen aanwezig, dan volstaat een melding.

show.tpl

```
{include file="header.tpl"}
{$gezocht}
{if $artikel != ""}
   <table border=2 cellpadding=0 cellspacing=0>
   <tr><td><b>Naam</b></td><td><b>Omschrijving </b></td>
       <td><b>Prijs in &euro;</b></td><td>Afbeeldingen</td></tr>
   {section name=mijnsectie loop=$artikel}
       <tr>
       <td><a href="winkelwagen.php?artikelid={$artikel[mijnsectie].artikelid}"
           >[bestel]</a>  {$artikel[mijnsectie].naam}</td>
       <td>{$artikel[mijnsectie].omschrijving}</td>
       <td>{$artikel[mijnsectie].prijs}</td>
       <td>
       {if $artikel[mijnsectie].images != ""}
           {foreach key=nr item=url from=$artikel[mijnsectie].images}
               <img src="{$url}"> 
           {/foreach}
       {else}
           Geen afbeeldingen voor dit artikel
       {/if}
       </td>
       </tr>
   {/section}
   </table>
{/if}
{include file="footer.tpl"}
```



Afbeelding 12.9 *Een artikeloverzicht in templates.*

Vragen en oefeningen

Beantwoord de volgende vragen zo nauwkeurig mogelijk.

1. Wat is het belangrijkste doel van het gebruik van templates?
2. Welke templatesystemen kent u voor PHP?
3. Noem een aantal belangrijke voordelen voor het gebruik van Smarty?

Oefeningen

Probeer de volgende oefeningen zo goed mogelijk uit te voeren.

1. Maak een aantal nieuwe Smarty-templates om de webshop een ander uiterlijk te geven.
2. Veel nieuwsbrieven hebben een vaste opzet. Maak een standaardnieuwsbrief (nieuwsbrieftemplate) met behulp van Smarty en geef de gebruiker de mogelijkheid zelf nieuws toe te voegen in de nieuwsbrief. Een nieuwsitem bestaat hierbij uit een titel, een inleiding en een body van de tekst. Sla zo nodig nieuwsitems op in de database.