2 Syntaxis van PHP

PHP en (X)HTML

Met PHP kun je van alles en nog wat programmeren. Alles dat je tot nu toe hebt gezien op het web, is te bouwen met PHP. Je kunt PHP gebruiken voor het verzenden van e-mail, het afhandelen van online betalingen, het bestandsbeheer van uploads en downloads enzovoort. Bij de meeste webapplicaties gebruik je PHP vroeg of laat echter voor het genereren van Hypertext Markup Language (HTML) of Extensible HTML (XHTML). De letter H in de afkorting PHP staat voor *hypertext*, dus daar zullen we dit hoofdstuk mee beginnen.

Lange PHP-tags

Je kunt op verschillende manieren server-side PHP-script insluiten in een webpagina. Er is maar één methode echt goed: open het script met de lange PHP-tag **<?php** en sluit het met **?>**. De hoofdstructuur en de *container* of *wrapper* van een PHP-script is altijd:

```
<?php
...
?>
```

Je mag de PHP-tags op één regel zetten en je mag ze insluiten in HTML-code. Heb je bijvoorbeeld een prijs in euro's opgeslagen in de PHP-variabele **$fPrijs**, dan kun je deze als volgt weergeven in HTML- of XHTML-code:

```
<p>Prijs: &euro; <?php echo $fPrijs; ?> inclusief btw.</p>
```

In PHP sluit je elke expressie af met een puntkomma. In dit geval mag je de puntkomma echter weglaten, omdat de eindtag **?>** ook dienstdoet als instructie voor het afsluiten van een PHP-expressie:

```
<p>Prijs: &euro; <?php echo $fPrijs ?> inclusief btw.</p>
```

Geen korte PHP-tags

In plaats van de lange PHP-tags **<?php ... ?>** kun je vaak de korte PHP-tags **<? ... ?>** gebruiken, met de starttag **<?** in plaats van **<?php**. Dat is echter af te raden. Met de directive **short_open_tag** in php.ini kunnen de korte PHP-tags worden uitgeschakeld. De lange PHP-tags kunnen niet worden uitgeschakeld en werken altijd. Met de korte PHP-tags wordt een PHP-script afhankelijk van de serverconfiguratie. De lange PHP-tags zijn dus de standaardtags en de korte PHP-tags zijn meer een optie die niet altijd werkt.

html-v1.php

```
<?php
// PHP-code in de lange PHP-tags
echo '<h1>Dit is PHP</h1>';
?>
<p>Dit is geen PHP maar iets anders,
bijvoorbeeld HTML of XHTML.</p>
```

De korte PHP-tags **<? ... ?>** waren oorspronkelijk bedoeld als *shorthand* of *steno* voor het snel 'inkloppen' van PHP-code. Tegenwoordig worden ze echter vooral gezien als gemakzucht van luie programmeurs. Zo heel veel werk is **<?php** in plaats van **<?** typen namelijk niet. Bovendien kun je in elke editor met 'zoeken en vervangen' **<?** vervangen door **<?php**. Kleine moeite.

PHP-script in kleur

In dit boek is alle broncode gedrukt in de officiële kleuren voor de *syntax highlighting* van PHP. Deze kleuren worden gebruikt in de officiële *PHP Manual* en op honderden websites over PHP. De standaardkleuren worden echter niet zomaar door iedereen van elkaar 'geleend'. Ze liggen vast in de *core* van PHP zelf:

highlight.default (#0000BB) Blauw is de standaardkleur voor PHP-code die geen speciale kleur krijgt. Blauw wordt onder meer gebruikt voor de PHP-tags **<?php** en **?>**, voor functies en voor variabelen.

highlight.keyword (#007700) Groen wordt gebruikt voor alle *keywords* of *sleutelwoorden* in php. Dit zijn de bouwstenen waarmee je onderdelen combineert, bijvoorbeeld operatoren, ronde haakjes, accolades en de puntkomma's voor het afsluiten van expressies.

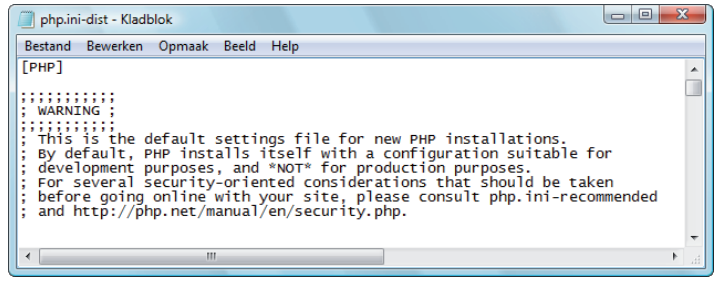
highlight.comment (#FF8000) Commentaar wordt weergegeven in oranje, een complementaire kleur van blauw. ⇢ Zie ook 'Commentaar' op pagina 53.

highlight.string (#DD0000) Rood wordt gebruikt voor strings. Een *string* is letterlijke tekst tussen enkele of dubbele aanhalingstekens.

highlight.html (#000000) Alle code buiten de PHP-tags wordt weergegeven in zwart. Meestal is dat HTML of XHTML. Als je HTML genereert met PHP, gebruik je daarvoor echter meestal strings en wordt de HTML-code dus rood weergegeven.

Hier lopen we via een omweg aan tegen een serieus beveiligingsprobleem. Elke distributie van PHP bevat twee versies van het configuratiebestand php.ini. Het bestand php.ini-dist bevat de flexibelere standaardconfiguratie voor een ontwikkelings- en testomgeving. Het bestand php.ini-recommended bevat een veiligere versie van php.ini met de aanbevolen configuratie voor een productieomgeving met live websites. En... in de veiligere versie php.ini-recommended is **short_open_tag** uitgeschakeld!

Als je merkt dat een live webserver de korte PHP-tags accepteert, kan dat heel toevallig liggen aan die ene instelling **short_open_tag**. In de praktijk blijkt er helaas meestal meer mis te zijn: de hostingprovider heeft een standaarddistributie van PHP geïnstalleerd, waardoor de onveilige versie van php.ini voor ontwikkelingsdoeleinden wordt gebruikt in plaats van de veilige versie voor productieservers. Als een server de korte tags accepteert, is dat helaas te vaak een veeg teken dat de provider de beveiliging niet op orde heeft en moet je daaraan zelf meer aandacht besteden.



*Als je PHP installeert met de standaardinstellingen, wordt php.ini-dist gebruikt voor de PHP-configuratie in php.ini. En de waarschuwing aan het begin van php.ini-dist zegt het al: deze versie van php.ini is bedoeld voor ontwikkelingsomgevingen en *NIET* voor productiedoeleinden!*

Output echoën met echo of print()

Als je binnen de PHP-tags output naar de client wilt verzenden, gebruik je **echo** gevolgd door de output. Het produceren van output met PHP wordt daarom vaak *echoën* genoemd. Bijvoorbeeld de PHP-stringvariabele **$sGebruikersnaam** kun je als volgt echoën om een gebruikersnaam weer te geven op een webpagina. Je vindt de volgende voorbeelden in html-v3.php tot en met html-v7.php in de download bij dit boek:

```
echo $sGebruikersnaam;
```

Wil je HTML-tags of HTML-tekst combineren met PHP-output, dan kun je strings tussen enkele of dubbele aanhalingstekens gebruiken voor de HTML-code. Vervolgens gebruik je de punt **.** om alles te

combineren tot één geheel. In PHP is een punt de tekstoperator voor het samenvoegen van strings:

```
echo '<p>Gebruikersnaam: ' . $sGebruikersnaam . '</p>';
```

Bevat de PHP-variabele $sGebruikersnaam bijvoorbeeld de string 'Administrator', dan krijg je hiermee de HTML-code:

```
<p>Gebruikersnaam: Administrator</p>
```

Wat veel PHP-gebruikers niet weten, is dat je echo kunt aanroepen met meerdere argumenten, gescheiden door komma's. Onbekend maakt onbemind, maar dit is ook goed in PHP:

```
echo '<p>Gebruikersnaam: ', $sGebruikersnaam, '</p>';
```

Als je dat handiger vindt, kun je in plaats van één echo met drie argumenten drie keer echo met één argument gebruiken:

```
echo '<p>Gebruikersnaam: ';
echo $sGebruikersnaam;
echo '</p>';
```

Uit de syntaxiskleur groen kun je afleiden dat echo geen functie is maar een *language construct* of *taalconstructie*. Een PHP-functie (blauw) geeft altijd een resultaat terug. Is dat resultaat geen concrete waarde, dan is het minimaal de booleaanse (logische) waarde true of false om te melden de functie goed of fout is uitgevoerd. Voor een taalconstructie (groen) zoals echo geldt dat niet. Een taalconstructie doet gewoon haar werk, zonder een resultaat te melden.

Wil je liever echoën met een functie, dan kun je print() gebruiken. Ook print() is een taalconstructie (*language construct*), maar dan een die je kunt gebruiken als functie. Het resultaat van print() is namelijk altijd de integer 1. In de praktijk heb je echter niets aan dat resultaat en kun je evengoed echo gebruiken. Het resultaat van bijvoorbeeld print($sGebruikersnaam) is namelijk ook 1 als de variabele $sGebruikersnaam niet bestaat of leeg is en er dus niets kan worden weergegeven. Omdat echo geen resultaat retourneert, is echo bovendien altijd iets sneller dan print().

XML en XHTML

Als de directive **short_open_tag** in php.ini is ingeschakeld, kan de PHP-parser zich verslikken in XML (Extensible Markup Language) en alle vormen van XML, waaronder XHTML (Extensible HTML). Je ziet dan een foutmelding zoals 'Parse error: syntax error, unexpected T_STRING in C:\...\hello-v5.php on line 4' bij het voorbeeldbestand

Tip! Technisch is er een groot verschil tussen echo met een punt en echo met een komma. De punt is de PHP-operator voor het samenvoegen van strings, de *concatenation operator* in het Engels. Als je een punt gebruikt, voert de PHP-engine eerst aparte operaties uit voor het samenvoegen van alle strings en wordt daarna pas het resultaat getoond met echo. Als je komma's gebruikt, kunnen deze aparte samenvoegoperaties worden overgeslagen en worden onmiddellijk alle argumenten getoond. Technisch is echo met komma's daarom net iets sneller dan echo met punten. Praktisch maakt het echter weinig uit. De snelheidswinst bij komma's is zó klein, dat deze geen rol speelt in de meeste webapplicaties.

Tip! Het uitschrijven van elke echo houdt de HTML-structuur van webpagina's overzichtelijk. In het volgende voorbeeld wordt de variabele $sGebruikersnaam getoond in een HTML-tabel. Dit voorbeeld (html-v7.php in de download) is nog vrij eenvoudig, maar je kunt je voorstellen dat grote webpagina's en complexe HTML-tabellen al gauw onoverzichtelijk worden als je ze niet uitschrijft met een aparte echo voor elke belangrijke bouwsteen:

html-v7.php

```
// Begin van de tabel
echo '<table>';
echo '<tbody>';
// Rij voor de gebruikersnaam
echo '<tr>';
echo '<td>Gebruikersnaam:</td>';
echo '<td>';
echo $sGebruikersnaam;
echo '</td>';
echo '</tr>';
// Einde van de tabel
echo '</tbody>';
echo '</table>';
```