## Flexible Website-Verwaltung mit PostNuke, Teil 2

# Xanthia-Themes erstellen

***von Axel Guckelsberger und Steffen Voß***

Der zweite Teil unseres Workshops widmet sich dem Layout der Beispielseite. Sie werden sehen, wie flexibel und einfach wartbar PostNuke hier sein kann. Neben der Struktur von Themes in PostNuke lernen Sie die Funktionsweise des Smarty-basierten Templatings kennen.

Im ersten Teil wurden bereits die Begriffe „Template“ und „Theme“ erläutert, ebenso wurden die beiden in PostNuke benutzten Smarty-Unterklassen vorgestellt: Xanthia für Theme- und Block-Templates sowie pnRender für Modul-Templates. Natürlich können Sie sich auch eine der frei verfügbaren Xanthia-Themes herunterladen, bei vielen Projekten sind jedoch ein eigenes Design sowie diverse Anpassungen notwendig. Daher wird im Folgenden illustriert, wie einfach sich ein beliebig gegebenes Layout in ein Xanthia-Theme umsetzen lässt. Des Weiteren werden Ihnen die Individualisierung von Modul-Templates sowie die Erstellung eigener Erweiterung mithilfe von Plug-ins näher gebracht.

### Die Geschichte

Waren Xanthia und pnRender bei PostNuke .726 noch Zusatzmodule, ist die Layout-Engine seit .750 fester Bestandteil der PostNuke-Distribution. Seither ist es durch die mit Smarty eingeführte Trennung zwischen Inhalt und Form möglich, das Aussehen von Theme, Blöcken und Modulen per Templates anzupassen, ohne dabei Gefahr zu laufen, dass die durchgeführten Änderungen bei einem Update überschrieben werden. Wer also Smarty schon von anderen Applikationen kennt, sollte sich leicht auf PostNuke einstellen können.

Da Xanthia aber auch der erste Versuch der PostNuke-Entwickler war, ein solches System umzusetzen, wurden einige Funktionen nicht ganz optimal umgesetzt – dazu kam, dass es einige Zeit gedauert hat, bis sich durchgesetzt hat, was man damit überhaupt machen kann. Viele Funktionen haben sich seither bewährt, andere wurden für die neue Xanthia-Version in der nächsten PostNuke-Version verworfen. Wir möchten Ihnen vor allem zeigen, welche Optionen Sie ignorieren sollten und wie Sie effektiv mit diesem System arbeiten können.

### Formsache

Xanthia ist der Teil, der sich um das Gesamtlayout (Theme) einer Seite kümmert. Die Administration können Sie vorerst weitestgehend außen vor lassen. Lediglich die folgenden Konfigurationsoptionen sollten Sie wie angegeben einstellen:

- Caching nutzen: *aus*
- Auf aktualisierte Templates prüfen: *ein*
- Erneutes Kompilieren der *Templates* erzwingen: *ein*
- Templates in die Datenbank schreiben: *aus*

Die ersten drei Einstellungen bewirken, dass die von Ihnen vorgenommenen Änderungen in den Templates beim nächsten Laden der entsprechenden Seite „bemerkt“ werden. Im produktiven Betrieb können Sie diese Punkte wieder zurückstellen, um die Performanz des Gesamtsystems zu verbessern. Die Idee, die Templates in der DB zu speichern und damit online editierbar zu machen, hat mehr Probleme als Nutzen bewirkt und sorgt an vielen Stellen für eine komplizierte Handhabung. Aus diesen Gründen wird diese Funktion in PostNuke .8 entfallen.

### Das Xanthia-Theme

Falls Sie noch kein Design für Ihre Seiten vor Auge haben, sollten Sie sich nun eines zum Ausprobieren suchen – zum Beispiel auf OSWD [1]. Diese Vorlage werden Sie jetzt in ein Xanthia-Theme umbauen. Im Prinzip müssen Sie dazu nur die Demo-Daten aus dem Layout entfernen und an die entsprechende Stelle die Aufrufe für die Inhalte aus PostNuke einbauen.

Die Themes befinden sind im Unterverzeichnis */themes/* und sie enthalten zwei Steuerdateien, die eigentlichen Layout-Templates, die Stylesheets, die Grafiken sowie alle angepassten Modul-Templates.

Als Basis eines neuen Layouts eignet sich am Besten ein vorhandenes Layout, das man Schritt für Schritt an die eigene Vorlage anpasst. Dazu nennen Sie einfach das Verzeichnis eines mitgelieferten Layouts um, zum Beispiel */themes/testLayout*.

Genau dieser Verzeichnisname ist der Name ihres Themes. Er muss in der Datei *xaninfo.php* eingetragen werden. Außerdem können Sie sich als Autor und die Angabe, ob das Theme auf HTML basiert oder die Ausgabe in validem XHTML erfolgt, hinzufügen.

Die andere Steuerdatei ist die *xaninit.php*, in welcher Farbplatten und vordefinierte Template-Zonen festgelegt werden. Diese Farben nutzt zum Beispiel pnForum, um in der Thread-Ansicht die Hintergrundfarbe zu wechseln. Sie können diese Datei zur Übung erst einmal so lassen und sich später mit ihren Tiefen beschäftigen.

Wechseln Sie in das Unterverzeichnis *templates*, dort befindet sich mit der Datei *master.html* ein wichtiges Template – es enthält den Grundaufbau des Layouts. Die meisten Themes beinhalten weiterhin bereits Variationen dieser Datei in *templates/modules/*. So sind *home.html* und *admin.html* für die Startseite und die Administration üblich. Diese Dateien sind insofern die Haupt-Templates, als dass die Ausgaben aller anderen Komponenten (Module, Blöcke, Hooks) dort integriert werden. Die Haupt-Templates sind somit der Punkt, an dem alles endet und die endgültige Seite zusammengestellt wird.

## Frisch ans Werk

Wenn Sie die *master.html* mit dem Texteditor öffnen und hineinschauen, fallen neben viel normalen HTML nur einige in *<!--[* und *]-->* eingebettete Ausdrücke auf. Dies sind Elemente, die von Smarty interpretiert und durch Inhalte bzw. Ausgaben ersetzt werden (siehe Kasten „Template-Terminologie"). Mit dem Template *master.html* haben Sie das Rohgerüst der Seite vor sich. Nun fehlt das CSS, welches sich unter *themes/ThemeName/style/style.css* befindet. Damit haben Sie alle benötigten Bestandteile zur Hand, um Ihr Layout direkt umzusetzen – als handelte es sich um eine einfache statische HTML-Seite. Lediglich die Smarty-Ausdrücke rücken an die Stelle der Inhalte.

Die Web-Templates von OSWD enthalten meist ebenfalls eine Stylesheet-Datei und das eigentliche HTML – diese beiden Dateien öffnen Sie ebenfalls im Editor. Als erstes sollten Sie die CSS-Definitionen aus der Vorlage ins Stylesheet des Themes übernehmen. Dann widmen wir uns dem eigentlichen Template: Den Head-Bereich des ursprünglichen PostNuke-Themes behalten Sie einfach, den Body löschen Sie. Nun kopieren Sie aus der Vorlage den Body in die *master.html* und ersetzen die Beispielinhalte durch die Aufrufe für die Blöcke und den Inhalt. Um herauszufinden, welche Variablen im aktuellen Template (egal ob Theme- oder Modultemplate) zur Verfügung stehen, können Sie einfach am Ende des Templates folgenden Aufruf hinzufügen:

```
<!--[ pndebug output="html" ]-->
```

Wenn Sie dann die Seite aufrufen, bekommen Sie als Admin eine Liste alle Variablen unter der normalen Seite eingeblendet.

Neben den bereits sehr mächtigen Werkzeugen, die Smarty beinhaltet, stellt Xanthia ein paar zusätzliche Variablen und Plug-ins zur Verfügung, von denen die wichtigsten kurz vorgestellt werden sollen (Tabelle 1 und 2).

Die meisten Plug-ins benötigen und akzeptieren verschiedene Parameter. Viele unterstützen etwa den Parameter *assign*, mit dem das Ergebnis des Plug-ins nicht ausgegeben, sondern einer Variable zugewiesen werden kann. Um die verfügbaren Parameter eines speziellen Plug-ins zu erfahren, öffnen Sie die gleichnamige Datei unter *modules/Xanthia/plugins/*. Dort erklärt in der Regel ein Kommentar die konkrete Benutzung.

## Templates updatesicher lagern

Wie im ersten Teil bereits kurz angesprochen, lassen sich Templates von Modulen im Theme lagern. Auf diese Weise sind Ihre Änderungen erstens gegen das Überschreiben durch Updates gesichert, zweitens können Sie pro Theme verschiedene Ansichten für Module und Blöcke gestalten. Module benutzen meist folgende Verzeichnisse für ihre Bestandteile:

- *modules/ModulName/pntemplates*: Templates
- *modules/ModulName/pnimages*: Bilder
- *modules/ModulName/pnstyle*: Stylesheets
- *modules/ModulName/pntemplates/plugins*: Plug-ins

All diese Elemente haben im Theme Vorrang gegenüber den Originalen und können somit leicht von Ihnen „überschrieben" werden. Auch bei eigenen Ergänzungen (zum Beispiel das Anlegen neuer Plug-ins für das Modul) ist das Theme die erste Wahl, damit alle selbst durchgeführten Arbeiten zentral bleiben. Die Pendants für die aufgelisteten Verzeichnisse im Theme lauten:

- *themes/ThemeName/templates/modules/ModulName: Templates*
- *themes/ThemeName/images/ModulName: Bilder*
- *themes/ThemeName/style/ModulName: Stylesheets*
- *themes/ThemeName/templates/modules/ModulName/plugins: Plug-ins*

Sollten die Templates im Modul weiter in Unterverzeichnissen liegen, müssen auch diese angelegt werden. Bei advProfile zum Beispiel liegen die Templates unter *modules/advProfile/pntemplates/default*, der analoge Ordner im Theme lautet dann *themes/ThemeName/templates/modules/advProfile/default*.

Auch die Templates von Modul-Blöcken können überschrieben werden, sie müssen dann in *themes/ThemeName/templates/blocks/BlockName/* liegen.

## Pimp my Module

Als erste Übung werden Sie nun Schritt für Schritt das Layout der News-Artikel verändern. Im ersten Teil des Workshops

| | |
|---|---|
| *<!–[$themepath]–>* | Pfad zum aktuellen Theme, für relative Links |
| *<!–[$imagepath]–>* | Pfad zum Bildordner des Themes, für relative Links |
| *<!–[$leftblocks]–>* | Inhalt der linken Blöcke |
| *<!–[$rightblocks]–>* | Inhalt der rechten Blöcke |
| *<!–[$centerblocks]–>* | Inhalt der mittleren Blöcke |
| *<!–[$maincontent]–>* | Inhalt der aktuellen Seite |

Tabelle 1: Wichtige Variablen

| | |
|---|---|
| *<!–[displaygreeting]–>* | Gibt eine Begrüßung aus |
| *<!–[footmsg]–>* | Gibt den in den Einstellungen definierten Footer-Inhalt aus |
| *<!–[search]–>* | Gibt eine Suchbox aus |
| *<!–[sitename]–>* | Gibt den Namen der Website aus |
| *<!–[userlogin]–>* | Gibt eine Loginbox aus |

Tabelle 2: Wichtige Plug-ins


Abb. 1: Standard-Formatierung eines News-Artikels in Pagesetter

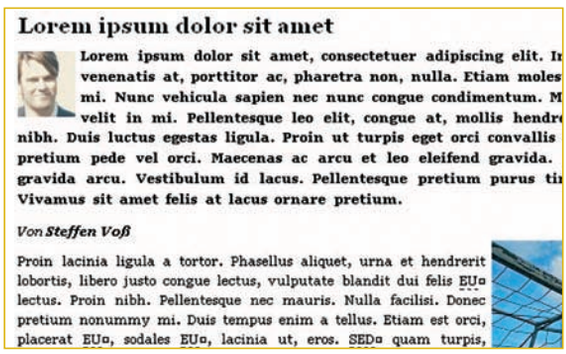


Abb. 2: Eine mögliche Alternative (hier: www.kaffeeringe.de)

hatten Sie das Modul Pagesetter installiert und ein, zwei News-Artikel damit erstellt. Falls Sie dies bisher noch nicht getan haben, laden Sie bitte Pagesetter [3] herunter. Ab der Version 6.3 ist schon ein Publication Type (Genaueres in Teil 3) für News und einfache Artikel enthalten. Um einige Inhalte zum Üben zu haben, sollten Sie einfach einige Artikel eingeben.

Die dazugehörigen Templates heißen *PN-News-*.html* und befinden sich unter *modules/pagesetter/pntemplates/*, von wo aus Sie sie direkt nach *themes/ThemeName/templates/modules/pagesetter/* kopieren. Wichtig für unsere Änderungen am Layout der News sind folgende Templates:

- *list* (*PN-News-list.html*): Anzeige eines Artikels innerhalb einer Listenansicht
- *list-header* (*PN-News-list-header.html*): Der Kopf einer Listenansicht
- *list-footer* (*PN-News-list-footer.html*): Der Fuß einer Listenansicht
- *full* (*PN-News-full.html*): Die Vollanzeige eines News-Artikels

Mit weiteren Templates lassen sich auch Druckversionen, RSS-, Atom- und sonstige Feeds oder alternative Ansichten erzeugen. Zu den vielseitigen Einsatzmöglichkeiten von Pagesetter erfahren Sie mehr im dritten Teil unseres Workshops.

Ihre News rufen Sie dann mit *index.php?module=pagesetter&tid=1* auf, die Vollansicht erreichen Sie indes unter *index.php?module=pagesetter&func=viewpub&tid=1&pid=1*, wobei pid die ID der anzuzeigenden Publikation darstellt. Um zu sehen, welche Variablen Sie außer den vorgegebenen verwenden können, benutzen Sie einfach wieder das pndebug-Plug-in, wie bereits gezeigt. Nehmen Sie vorerst einfach zwei oder drei Änderungen an den diversen Templates vor, um sich vor Augen zu führen, über wie viel Freiheiten Sie jetzt schon in Bezug auf Ihr Seitendesign verfügen. Zum Beispiel könnten Sie das mitgelieferte Tabellen-Layout in ein tabellenfreies verwandeln. Die beiden Screenshots illustrieren, wie unterschiedlich PostNuke-Seiten mit Xanthia aussehen.

## Plug-ins

Sie haben bereits einige Plug-ins kennen gelernt. Diese sind, wie bereits beschrieben, Funktionen, die von Templates aus aufgerufen werden. Damit trennt Smarty nicht nur Inhalt und Form, sondern – genauer betrachtet – auch Programmlogik und Ausgabelogik. Was bedeutet diese Trennung? Als API-Logik sind alle für die Funktionalität eines Modules unmittelbar nötigen Operationen zu betrachten, zum Beispiel die Datenschicht, welche die DB-Sätze den darauf zugreifenden Schichten zur Verfügung stellt. Die GUI-Logik sind dagegen Funktionen, deren Logik ausschließlich für die Darstellung relevant ist. Ein Beispiel dafür ist etwa das Durchlaufen einer Liste, um eine *<select>*-Auswahlliste zu erzeugen. Übrigens könnte man dieses Beispiel auch ohne ein extra Plug-in umsetzen:

```
<!--[foreach item=currentElem from=$myarray]-->...
                                    <!--[/foreach]-->
```

Allerdings sollte so viel Logik wie möglich aus den Templates herausgehalten werden. Der Grund ist einfach, dass die Templates auch von reinen Web-Designern bearbeitet können werden sollen. Eine Schleife oder ein paar Bedingungen sind da an sich nichts Schlimmes, allerdings sollte auch sichergestellt sein, dass die Logik einer Applikation sich nicht durch Änderungen in den Templates „zerschießen“ lässt.

### Template-Terminologie

Smarty-Templates beinhalten neben Markup-Elementen verschiedene andere Ausdrücke, die hier kurz vorgestellt werden sollen:

- Als erstes sind Variablen zu nennen, die durch ein *$* am Anfang kenntlich gemacht werden (Beispiel: *<!--[$title]-->*). Array-Elemente werden hierbei durch einen Punkt adressiert (*<!--[$content.title]-->*).
- Jeder Ausdruck, der nicht mit einem Dollarzeichen beginnt, ist entweder ein Steuerbefehl (für Bedingungen und Schleifen) oder aber ein so genanntes Plug-in. Dies ist nichts anderes als eine normale Funktion, nur mit dem Unterschied, dass sie nicht von PHP aus, sondern im Template aufgerufen wird. Diese Funktionen können Sie leicht selbst schreiben und somit individuelle Ausgaben an beliebiger Stelle im Template ermöglichen.
- Zusätzlich gibt es Modifier, mit denen Variablen noch im Template verändert werden können. Diese werden durch Pipe-Zeichen repräsentiert und lassen sich stapeln (*<!--[$title|upper|truncate:40:"..."]-->*). Außerdem können alle PHP-Funktionen als Modifier benutzt werden. Diese werden dann durch ein @ markiert. <--[$meinArray|@count]--> gibt zum Beispiel die Anzahl der Array-Elemente aus.

Die Lektüre des Smarty-Handbuches [2], welches auch als PDF zur Verfügung steht, ist Voraussetzung für ein effizientes Arbeiten mit Templates.

Plug-ins für die Ausgabelogik kommt in einem Theme eine spezielle Bedeutung zu. Wann immer Sie eine individuelle Logik im Rahmen des Aussehens benötigen oder implementieren, bietet sich ein Theme-Plug-in dafür an, welches unter *themes/ThemeName/plugins* gespeichert wird.

Ein konkretes Beispiel wird Ihnen das Gesagte schnell verdeutlichen. Wir wollen ein Plug-in bauen, was per Zufall eines von zehn vorhandenen Header-Grafiken anzeigt. Kopieren Sie die Datei *modules/Xanthia/plugins/function.charset.php* nach *modules/ThemeName/plugins/function.randomheader.php* und öffnen Sie diese im Editor. Unter dem Datei-Header sehen sie den erwähnten Kommentar zur Benutzung des Plug-ins. Darunter befindet sich die eigentliche Funktion aus Listing 1.

Ein Plug-in bekommt per Standard zwei Parameter: *$params* ist ein Array mit allen Parametern, die an das Plug-in weitergereicht werden, *&$* ist eine Referenz zu dem Smarty-Objekt, welches gerade aktiv ist. Die letzten vier Zeilen sind für die Unterstützung des assign-Parameters, welcher zuvor angesprochen wurde. Editieren Sie nun die Funktion wie in Listing 2.

Wenn die Bilder im angegebenen Ordner existieren und Sie dieses Plug-in nun im Template mit *<!--[randomheader]-->* aufrufen, sehen Sie das Ergebnis. Wenn Sie für eine weitere Verarbeitung das Ergebnis einer Variable zuweisen wollten, wäre der Aufruf *<!--[randomheader assign="varName"]-->* passend. Egal, ob Sie nun verschiedene Stile je nach Tageszeit einbinden oder abhängig von individuellen Bedingungen eine Werbe-Integration einblenden möchten – ein Theme-Plug-in macht es möglich, ohne irgend etwas am Grundsystem von PostNuke verändern zu müssen.

Natürlich können Sie an solchen Stellen auch sämtliche Funktionen von PostNuke und den installierten Modulen benutzen. So können Sie ohne Hacken der Modul-Dateien Funktionen nachrüsten oder Funktionen andere Module kreuz und quer nutzen – und das Ganze ist auch noch sicher vor Updates. Näheres zu den Programmierschnittstellen des pnAPIs wird der dritte Teil des Workshops erklären.

**Listing 1**

```
function smarty_function_charset($params,
                                 &$smarty)
{
  $return ='';
  if (defined('_CHARSET')) {
        $return = _CHARSET;
  }

  if (isset($params['assign'])) {
    $smarty->assign($params['assign'], $return);
  } else {
    return $return;
  }
}
```

**Listing 2**

```
function smarty_function_randomheader
                        ($params, &$smarty)
{
  $return ='';
  $randomNumber = mt_rand(0, 9);
  $return = 'themes/ThemeName/images/
    headerpics/header_' . $randomNumber . '.jpg';

  if (isset($params['assign'])) {
    $smarty->assign($params['assign'], $return);
  } else {
    return $return;
  }
}
```

## Was noch?

Eine weitere Funktion von Xanthia ist die Block-Kontrolle, mit der Sie Zuordnungen zwischen Modulen und Blöcken definieren können. So lässt sich z.B. der Block mit den neuesten Einträgen im Forum nur im Forum und im Gästebuch anzeigen. Die Praxis hat gezeigt, dass die schöne Idee der Blockkontrolle an einer unflexiblen Verwaltung scheitert. Abgesehen davon ist es ein etwas unperformanter Ansatz, eher zu empfehlen ist es mit dem pnRender-Plug-in *pnblock* einzelne Blöcke an beliebiger Stelle einzubauen und deren Anzeige im Template von einer Bedingung abhängig zu machen.

Ein immer essenzieller werdender Punkt ist die Suchmaschinenfreundlichkeit eines Auftrittes. Neben der freien Anpassbarkeit des Markups rücken dabei zwei weitere Aspekte in den Vordergrund: erstens ein dynamisches *title*-Tag auf den Einzelseiten, zweitens *short_urls*. Falls *mod_rewrite* auf dem Server vorhanden ist, unterstützt Xanthia die schöneren URLs direkt durch einen Smarty-Outputfilter, welcher in der kompletten Ausgabe einer Seite alle Links durch die kürzere Form ersetzt. Xanthia bietet dabei die Wahl zwischen drei Endungen an (htm, html, phtml). Um die *short_urls* zu aktivieren, müssen Sie lediglich folgende Schritte durchführen:

- Die Datei */modules/Xanthia/pndocs/short_urls/EndungIhrerWahl/.htaccess* ins Wurzelverzeichnis der PostNuke-Installation kopieren.
- In der Administration von Xanthia die short_urls aktivieren.

Bei dem anderen Punkt (dynamische Titel) muss Xanthia leider passen. Innerhalb von Pagesetter gibt es das Plug-in *pagesetter_title*, welches das Setzen eines seitenspezifischen Titels vom Template aus erlaubt. Eine noch mehr Module umfassende Lösung existiert mit dem TitleHack [5]. Dieser sucht in den Modulverzeichnissen nach einer *pnTitle.php*, inkludiert diese und führt eine spezielle Funktion aus, um den Titel der aktuell aufgerufenen Seite zu ermitteln. Die Anwendung ist allerdings denkbar einfach: Es bestehen schon für eine Reihe von Modulen fertige Dateien, die Sie einfach in die Modulverzeichnisse kopiert werden müssen. Für die Integration steht auch hier ein Plug-in zur Verfügung, welches einfach das in Xanthia per Standard mitgelieferte *title*-Plug-in. Xanthia bietet darüber hinaus noch eine Reihe weiterer Features, wie etwa Theme-Variablen, selbst definierte Theme- und Block-Zonen oder dynamisch generierte Stylesheets. Hierzu sei auf das Handbuch für Xanthia und pnRender verwiesen [6].


Beide Autoren sind im Postnuke e.V. als Entwickler und Organisatoren für die Community tätig. Sie erreichen sie per Mail unter info@guite.de (Axel) bzw. kontakt@kaffeeringe.de (Steffen).

### Links & Literatur


[1] Open Source Web Design: www.oswd.org

[2] Handbuch für Smarty: smarty.php.net/manual/de/

[3] Pagesetter: www.elfisk.dk

[4] News erstellen mit Pagesetter: www.elfisk.dk/modules/pagesetter/docs/manual/PagesetterManual.html#chap_tutorial

[5] TitleHack: sourceforge.net/projects/lottasophie/

[6] Handbuch für Xanthia & pnRender: support.pn-cms.de/modules/dokuwiki/xanthia/index