Inhalt

Umgebung, Defaults

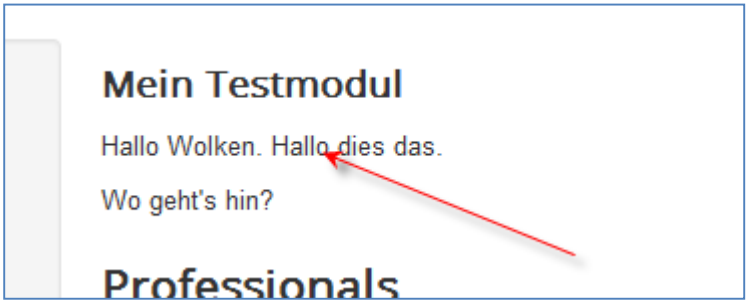
Testumgebung: Joomla 3.4.3, Template Protostar.

Wie immer ist empfohlen, sich eine Kopie des Protostar-Templates anzulegen, wenn man das originale nicht kaputt machen möchte. http://forum.joomla.de/index.php/Thread/315-Joomlaeigene-Templates-anpassen-%C3%A4ndern-z-B-Protostar-Beez3-Vorher-eine-Template-/

Modul vom Typ *Eigene Inhalte* anlegen

Lege ein neues Modul vom Typ *Eigene Inhalte (Leeres Modul)* in deinem Joomla an, gebe etwas Text in den Editor ein, stelle *Titel anzeigen* auf `JA`, weise *Position* `position-3` zu und überprüfe, ob es im Frontend angezeigt wird, nachdem du unter *Menüzuweisung* `Auf allen Seiten` eingestellt hast.

Erst mal egal, wie es aussieht und ob der Titel angezeigt wird. Bei mir schon, was an der Position `position-3` liegt, die im Protostar-Template eine Überschrift vorsieht:



Vorhandene Modulstile verwenden

Warum sehe ich oben eine Überschrift?

Wechsele in den Moduleinstellungen in den Tabulator *Erweitert* und siehe die Einstellung im Feld *Modulstil*: `Vererbt`.

Von wem geerbt?

Vom aktiven Template. Werfe ich einen Blick in die index.php meines aktiven Protostar-Template, suche nach `position-3`, finde ich u.a. diese Zeile (bei mir 176):

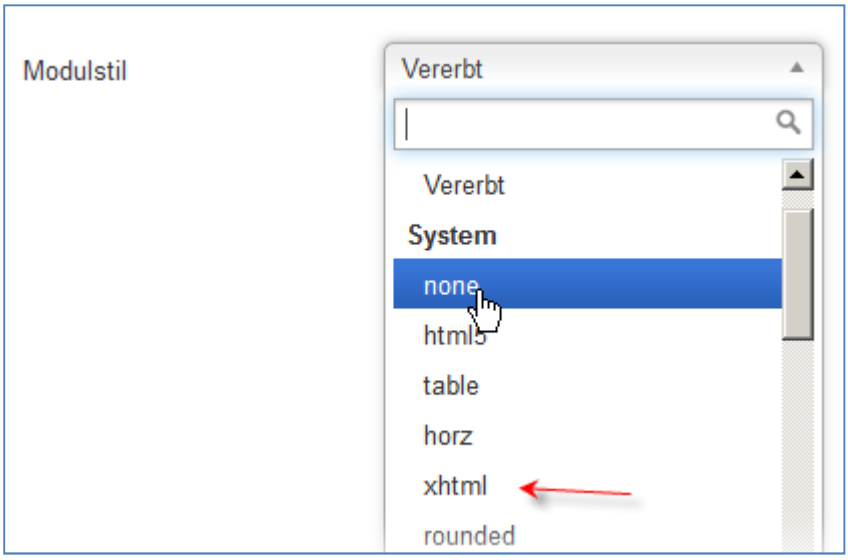
```
<jdoc:include type="modules" name="position-3" style="xhtml" />
```

Mit dieser Zeile zeigt das Template ein oder mehrere Module (`type="modules"`) an, die in der Positionsauswahl den Positionsnamen `position-3` haben.

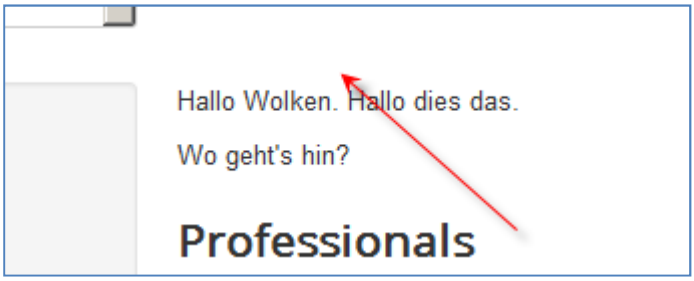
Und dann gibt es noch dieses `style="xhtml"` in der Zeile. Das ist der Modulstil, den das Template "empfiehlt" für diese Position.

Stilvererbung überschreiben:

Ändere nun den *Modulstil* im Backend von `Vererbt` auf `System > none` und speichere das Modul.



Überschrift ist weg. **Trotz Moduleinstellung *Titel anzeigen* auf `JA`**:



Alle Modulstile unter der Überschrift System, sind welche, die Joomla mitbringt, stehen also in allen Templates zur Verfügung. Dort siehst du auch den oben von Protostar "empfohlenen" Stil `xhtml`.

Wenn du im Auswahlfeld weiter schaust, siehst, dass man auch Modulstile anderer Templates einstellen kann.

Zwischenspielerei:

Wähle den Stil `System > html5` und speichere. Du wirst feststellen, dass jetzt auch andere Modul-Einstellungen wie z.B. *Header-Tag* eine Wirkung auf das angezeigte Modul haben, was bei vielen anderen Stilen nicht der Fall ist.

Eigene Ausgabezeilen in Modulstil (Vorübungen)

Zurückschalten zum Stil `none`.

Hinweis vorab: In den Dateien im Ordner `/templates/system/` ändert man (eigentlich) nie was!!!! Trotzdem bin ich jetzt mal zur Einleitung so frei.

Wirf einen Blick in die Datei `/templates/system/html/modules.php`

Ziemlich am Anfang diese Zeilen:

```
function modChrome_none($module, &$params, &$attribs)
{
 echo $module->content;
}
```

Wir endecken die Stilbezeichnung `none` in der ersten Zeile, davor ein `modChrome_`. So findet Joomla diesen Stil. Es sucht in allen Unterordnern von `/templates/` und dort im jeweiligen Unterordner `/html/` nach einer

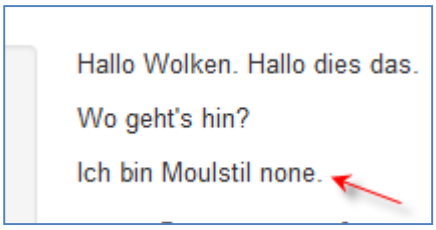
Datei `modules.php`. Sammelt alle `modChrome_xyz` zusammen und zeigt bspw. im Backend das `xyz` als Modulstil zur Auswahl an.

Gebe spaßeshalber eine weitere Zeile innerhalb der geschweiften Klammern von `function modChrome_none` ein:

```
echo "Ich bin Moulstil none.";
```

speichere die Datei, schau im Frontend.

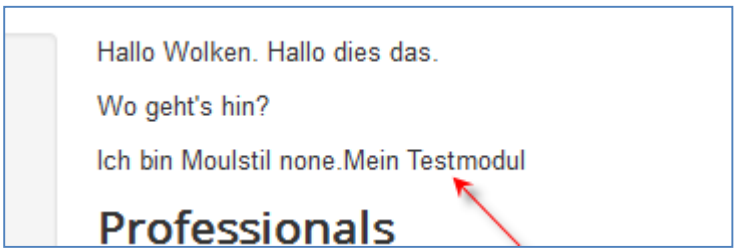
Alle Module mit Stil `none`, egal, welche Position, welcher Modultyp, welches Template bekommen jetzt deine Extrazeile angezeigt.



Und der Modultitel wird NICHT angezeigt, egal, was man im Backend eingestellt hat, da entsprechende Ausgabezeilen in diesem Modulstil `none` fehlen.

Also füge eine weitere Zeile innerhalb der geschweiften Klammern ein:

```
echo $module->title;
```



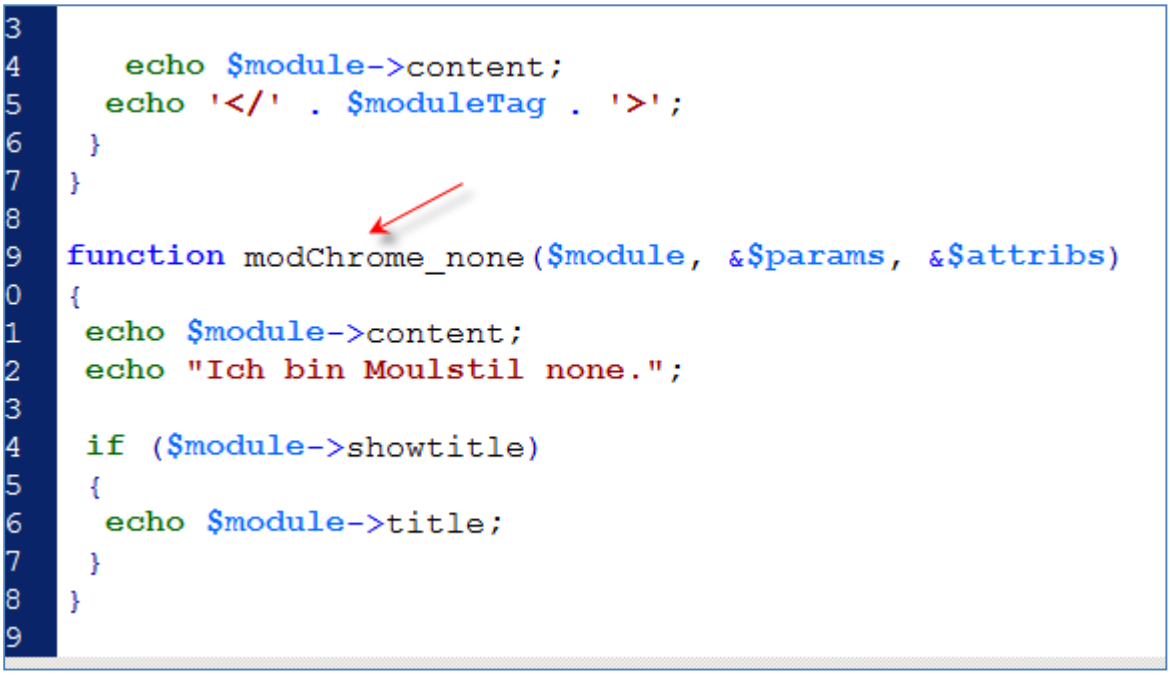
Blöd nur, dass die Überschrift jetzt immer angezeigt wird, egal, was man im Modul unter *Titel anzeigen* einstellt. Also müssen wir die Einstellung `showtitle` auch noch berücksichtigen, in einer `if`-Schleife (eigentlich if-Condition, egal). Mein Code sieht dann so aus:

```
function modChrome_none($module, &$params, &$attribs)
{
 echo $module->content;
 echo "Ich bin Moulstil none.";

 if ($module->showtitle)
 {
  echo $module->title;
 }
}
```

Nicht vergessen: Entferne die selbst eingefügten Zeilen wieder und speichere die Datei bevor wir sie verlassen.

Eigenen Modulstil anlegen

Danach öffne ich die Datei meines Templates Protostar `/templates/protostar/html/modules.php` und füge den obigen Code ganz am Ende der Datei ein, hinter der letzten geschweiften Klammer. Speichere die Datei.



```
3
4       echo $module->content;
5     echo '</' . $moduleTag . '>';
6   }
7 }
8
9 function modChrome_none($module, &$params, &$attribs)
0 {
1  echo $module->content;
2  echo "Ich bin Moulstil none.";
3
4  if ($module->showtitle)
5  {
6   echo $module->title;
7  }
8 }
9
```

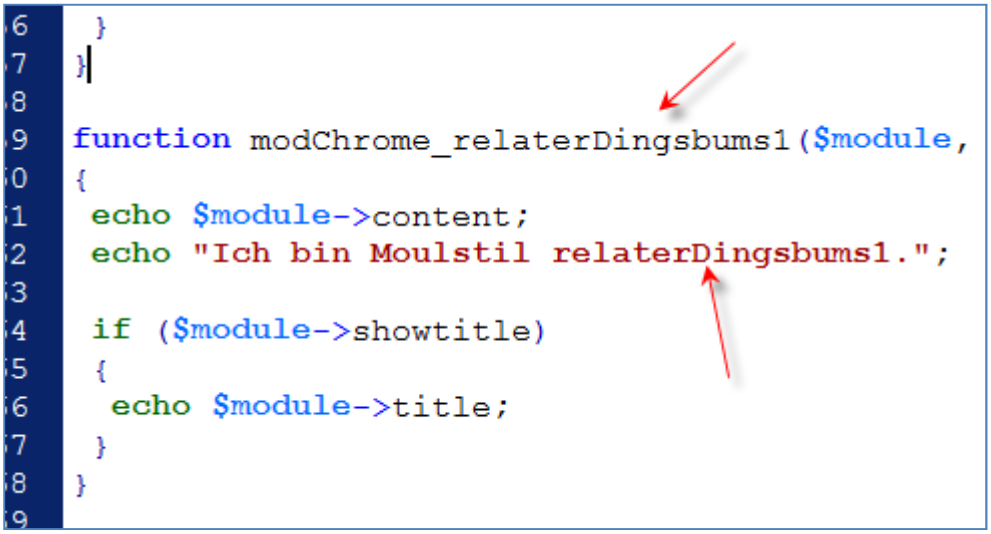
Juhu, mein eigener Modulstil.

Denkste, die Seite hängt sich auf. Weißes Frontend oder diese Meldung, wenn du *Fehler berichten* in der Joomla-Konfiguration entsprechend eingestellt hast:

Fatal error: Cannot redeclare modChrome_none() (previously declared in /templates/system/html/modules.php:15) in **/templates/protostar/html/modules.php** on line

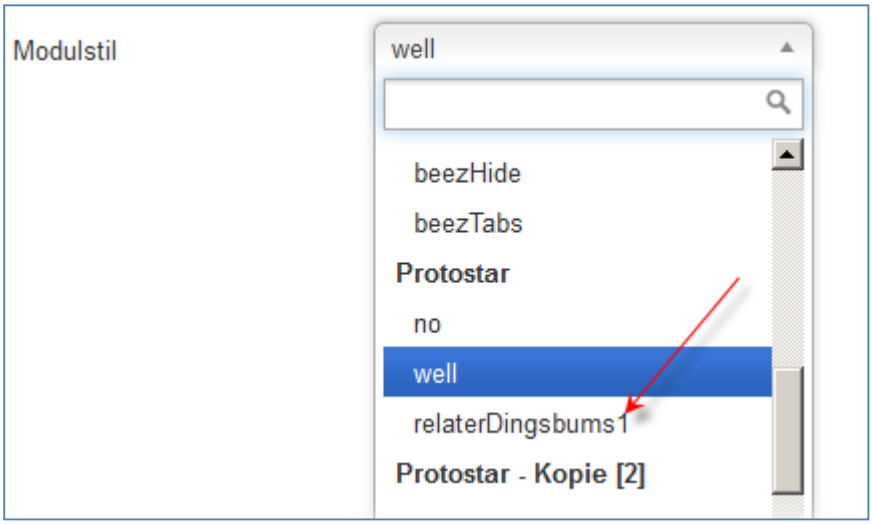
"Einen Modulstil namens none gibt es schon. Ich kann mit einem zweiten mit selbem Namen nichts anfangen. Tschüss!"

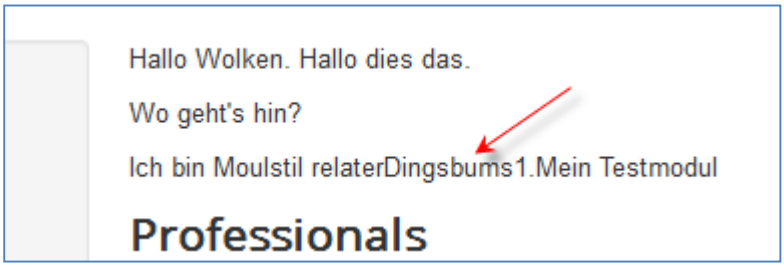
Hilft unverwechselbar(!) umbenennen. Ich leite meine Stilnamen alle mit einem `relater` ein. So lassen sie sich auch leichter mit dem Suchfeld in der Backend-Modulstilauswahl als Gruppe finden. Also wie's beliebt, aber keine Leerzeichen, Umlaute, Bindestriche und immer ein `modChrome_` vorne dran.



```
6   }
7 }
8
9 function modChrome_relaterDingsbums1($module,
0 {
1  echo $module->content;
2  echo "Ich bin Moulstil relaterDingsbums1.";
3
4  if ($module->showtitle)
5  {
6   echo $module->title;
7  }
8 }
9
```

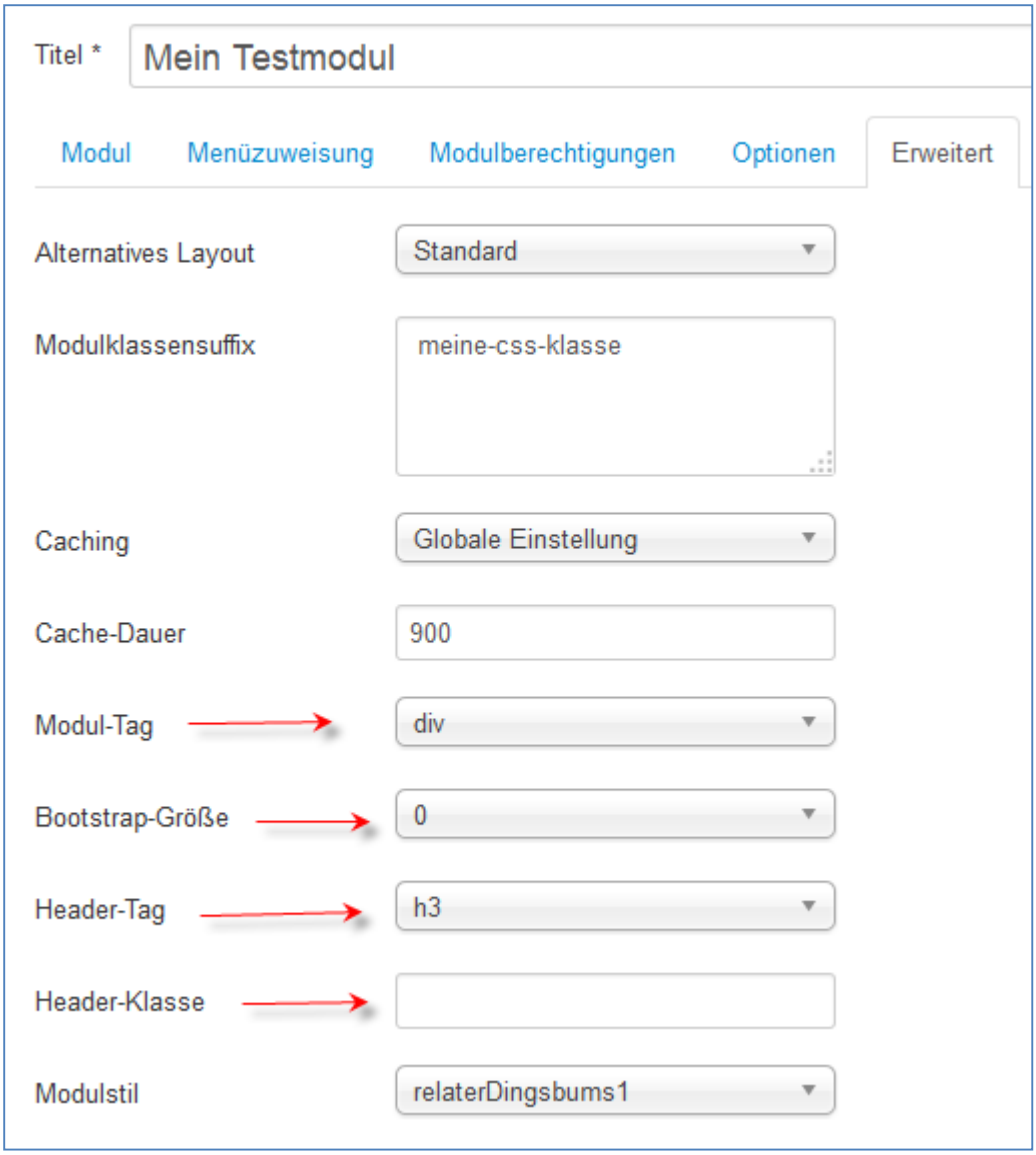
Frontend lädt jetzt wieder fehlerfrei und im Backend kann ich den eigenen Stil nun auswählen (nach neu laden des Modulfensters, damit die Auswahlbox aktualisiert wird).





Modul-Parameter im Modulstil verwenden

In den Moduleinstellungen *Erweitert* gibt es ja jetzt noch weitere Parameter, die man einstellen kann, aber in unserem Modulstil bisher keine Wirkung haben (rote Pfeile im folgenden Bild).



Parameter-Namen in Erfahrung bringen:

Um zu wissen, wie die intern, also codetauglich bezeichnet sind, kann man z.B. in die Datei

`/administrator/components/com_modules/models/forms/advanced.xml`

schauen.

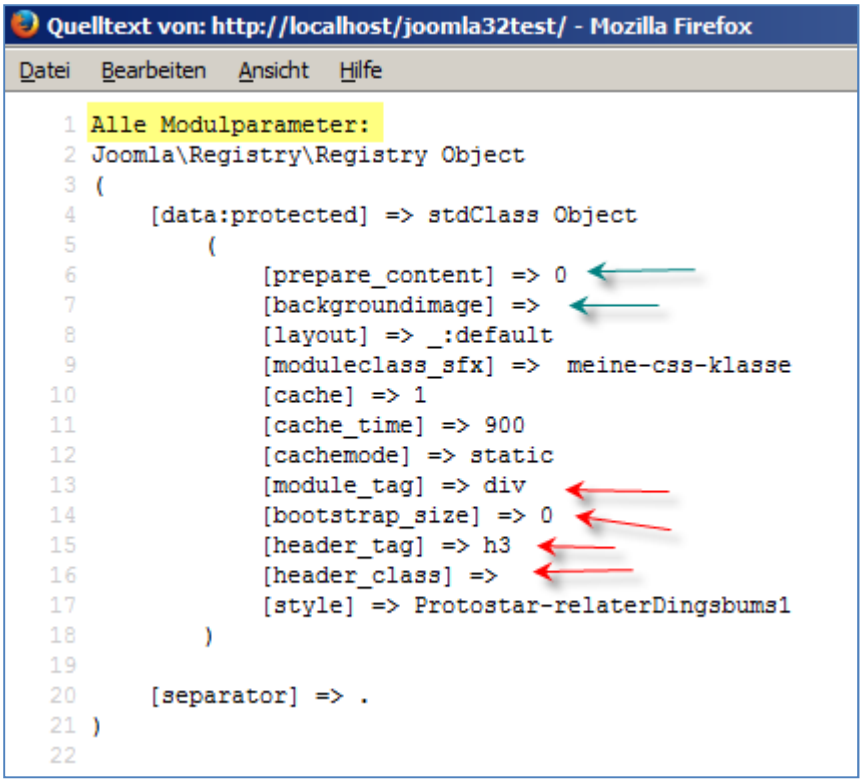
```
<fieldset
 name="advanced">

 <field
  name="module_tag"
  type="moduletag"
  label="COM_MODULES_FIE
  description="COM_MODUL
  default="div"
 />

 <field
  name="bootstrap_size"
  type="integer"
  first="0"
  last="12"
  step="1"
  label="COM_MODULES_FIE
  description="COM_MODUL
 />

 <field
  name="header_tag"
  type="headertag"
  default="h3"
```

Nachdem es aber sein kann, dass die Datei irgendwann nicht mehr an dieser Stelle ist, hole ich mir alle verfügbaren Parameter via PHP-Code und sehe mehr als in der XML-Datei.

```
echo "Alle Modulparameter:\n" . print_r($params, true);exit;
```

```
function modChrome_relaterDingsbums1($modu
{
 echo "Alle Modulparameter:\n" . print_r($

 echo $module->content;
 echo "Ich bin Moulstil relaterDingsbums1.

 if ($module->showtitle)
 {
```

Im Frontend jetzt etwas unübersichtlich die Ausgabe auf einer nahezu weißen Seite. Öffne deshalb den HTML-Quelltext der Seite:

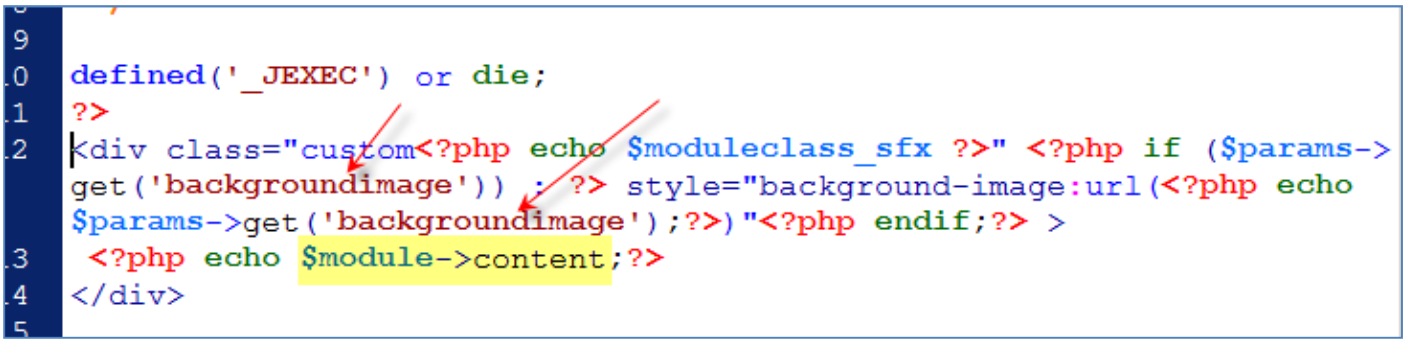


Man sieht je nach Modultyp mehr oder weniger Parameter, die nur in diesem Modul existieren (blaue Pfeile). Das Modul *Eigene Inhalte* hat ja Einstellungen *Inhalt vorbereiten* (`prepare_content`) und *Hintergrundbild* (`backgroundimage`), die in anderen Modultypen nicht (immer) existieren.

Danach ein paar Standard-Parameter (ohne Pfeile), die zwar nicht in allen Modulen vorkommen MÜSSEN, aber meist doch vorhanden sind. Mich nerven Module extrem, die bspw. keine Auswahl *Alternatives Layout* (Parameter `layout`) haben... Mal so nebenbei...

Und die rot bepfeilten sind aus unserer XML-Datei oben.

Diese Parameter kannst du in deinem Stil alle verwenden, alle auslesen. Oft werden diese aber anderweitig schon verwendet/berücksichtigt. Siehe z.B. den Parameter `backgroundimage` in der Datei `/modules/mod_custom/tmpl/default.php`



Würde man in seinem Stil das `backgroundimage` ebenfalls verwenden... Kann gut aussehen, muss aber nicht.

Und weiterhin machen sie in Modulen, in denen es die Parameter gar nicht gibt, nicht immer Sinn, wenn man sie im Stil verwendet. Das wird gleich gezeigt, wie man einen voreingestellten Wert verwenden kann, wenn Nutzer gar nichts gewählt hat oder Parameter gar nicht im Modul einstellbar ist.

Als erstes der Parameter `header_class` (Header-Klasse), weil der bei mir in den Moduleinstellungen noch leer ist. Wenn der User was eingetragen hat, soll sein Wert genommen werden, wenn nicht, dann ein im Stil voreingestellter Wert genommen werden. Die Überschrift `$module->title` unseres Codes bekommt ein `class="..."` hinzu, damit wir sie per CSS individuell formatieren können.

Dafür lese ich den Parameter-Wert wie folgt aus und, wenn er leer ist, wird die Klasse `meineEigeneKlasse` genommen. Das Ergebnis wird zur Weiterverwendung in der PHP-Variablen `$headerClass` abgelegt.

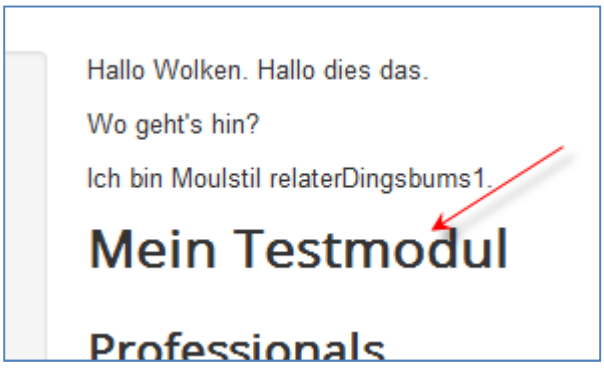
```
$headerClass = $params->get('header_class', 'meineEigeneKlasse');
```

und setze dann diese CSS-Klasse in meinen HTML-Tag ein, mit dem ich den Titel umgeben habe. Für den Moment ein H1. Mit den Punkten verknüpft man Strings ("Texte") in PHP.

```
echo '<h1 class="' . $headerClass . '">' . $module->title . '</h1>';
```

```
58
59  function modChrome_relaterDingsbums1($module, &$params,
60  {
61   $headerClass = $params->get('header_class', 'meineEige
62
63   echo $module->content;
64   echo "Ich bin Moulstil relaterDingsbums1.";
65
66   if ($module->showtitle)
67   {
68    echo '<h1 class="' . $headerClass . '">' . $module->t
69   }
70  }
```

Frontend:



Blick in den HTML-Quelltext der Seite (ich habe ihn etwas gehübscht:

```
<div class="custommeine-css-klasse"  >
 <p>Hallo Wolken. Hallo dies das.</p>
 <p>Wo geht's hin?</p>
</div>
Ich bin Moulstil relaterDingsbums1.
<h1 class="meineEigeneKlasse">Mein Testmodul</h1>
```

H1 hat jetzt die im Stil voreingestellte `class`, da keine im Backend eingegeben wurde.

Beachte, dass der DIV-Container aus der oben gezeigten `default.php` des Moduls kommt und die Ausgaben des eigenen Stils außerhalb des DIV liegen. Erst mal blöd.

Das `$module->content` in unserem eigenen Stil-Code hat also die moduleigene, oben gezeigte `default.php` bereits durchlaufen, bevor der aufbereitete Modulinhalt (`content`) im Stil ergänzt wird.

Jetzt führe ich noch eine Technik ein, die MIR im Moment (kann morgen schon wieder anders sein) das zusammenstellen der Zeilen etwas übersichtlicher macht. Wenn du dir die diversen Stile des sytems-Template und anderer Templates anschaust, siehst du, dass das auch anders geht. Auch im sonstigen Joomla-Code findest du vielleicht für dich angenehmere Alternativen, die man auch mischen kann.

```
function modChrome_relaterDingsbums1($module, &$params, &$attribs)
{
 $headerClass = $params->get('header_class', 'meineEigeneKlasse');

 $Regal = array();

 if ($module->showtitle)
 {
  $Regal[] = '<h1 class="' . $headerClass . '">';
  $Regal[] = $module->title;
  $Regal[] = '</h1>';
 }

 $Regal[] = $module->content;
 $Regal[] = '<p class="small">Ich bin Moulstil relaterDingsbums1.</p>';

 $Regal = implode('', $Regal);

 echo $Regal;
}
```

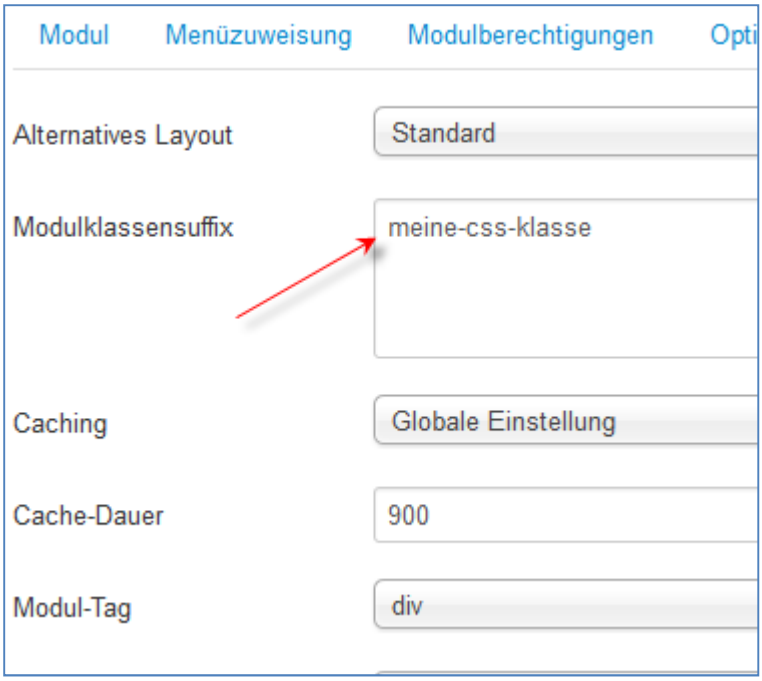
Erst lege ich ein PHP-Array an, quasi ein Regal, weshalb ich es `$Regal` nenne, in das man Text ein/umsortieren kann.

Da lege ich schrittweise meine Zeilen rein, u.a. auch mein `$module->content`. Die eckigen Klammern. Beim `$module->title` habe ich die Zeile sogar in 3 Fragmente zerlegt, öffnenden H1-Tag, Titel-Text, schließenden H1-Tag.

Am Ende fasse ich die Zeilen wieder in einen Text zusammen. Das `implode`.
http://php.net/manual/de/function.implode.php

Und gebe den zusammengefassten Text per `echo` aus, statt für jede Code-Zeile ein eigenes `echo` zu verwenden.

Wie wir oben festgestellt haben, liegen unsere zusätzlichen Texte aus dem Stil außerhalb des DIV-Containers. Unser Modulklassensuffix (auch Modulklassenpräfix genannt), kann also mittels CSS nicht auf diese Texte angewendet werden.



Ein Blick in die Parameterliste oben zeigt, dass der Parameter `moduleclass_sfx` noch im Stil eingebaut werden sollte, so dass er die gesamte Modulausgabe umschließt. Diesmal soll jedoch der umschließende DIV-Container, wenn der Benutzer keine Eingabe gemacht hat, auch kein `class="..."` bekommen.

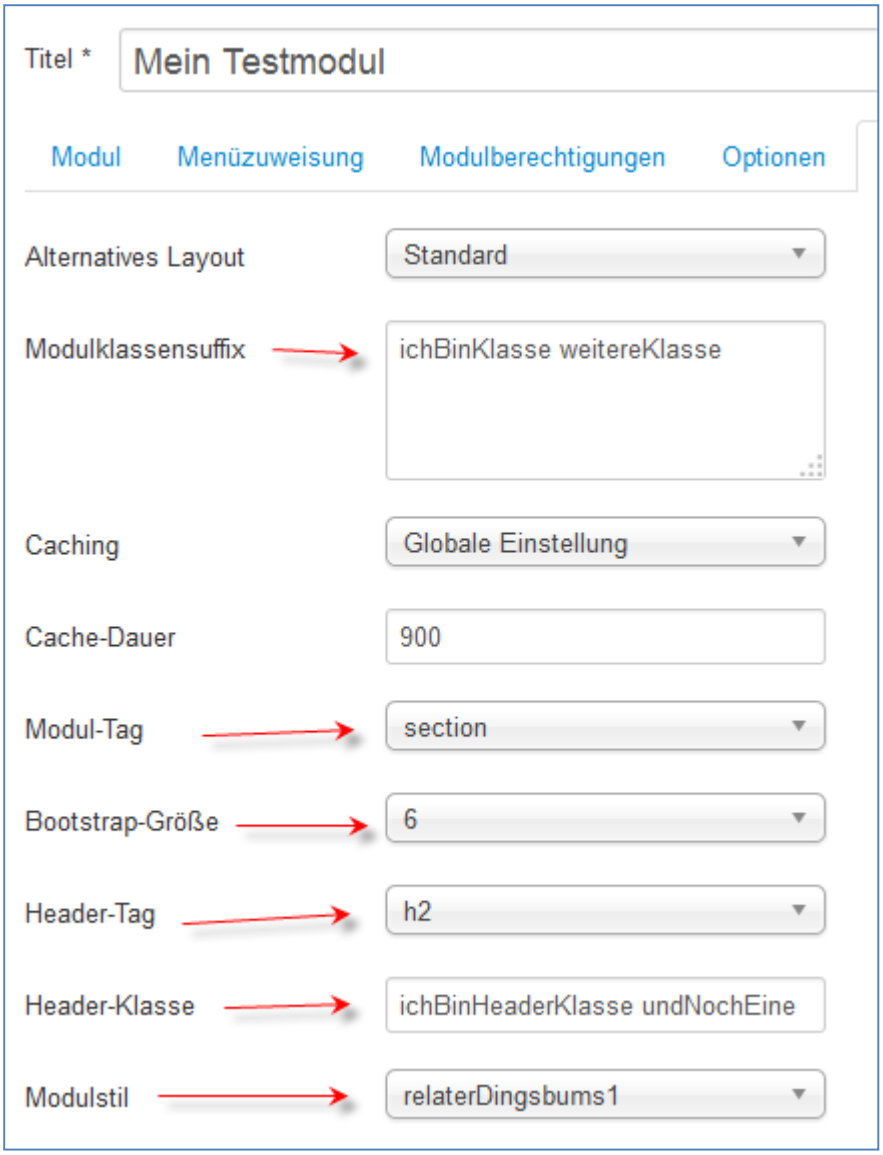
```
function modChrome_relaterDingsbums1($module, &$params, &$attribs)
{
 $headerClass = $params->get('header_class', 'meineEigeneKlasse');
 $moduleClass = $params->get('moduleclass_sfx', '');

 $Regal = array();

 $Regal[] = '<div';
 if ($moduleClass)
 {
  $Regal[] = ' class="' . $moduleClass . '"';
 }
 $Regal[] = '>';

 if ($module->showtitle)
 {
  $Regal[] = '<h1 class="' . $headerClass . '">';
  $Regal[] = $module->title;
  $Regal[] = '</h1>';
 }

 $Regal[] = $module->content;
 $Regal[] = '<p class="small">Ich bin Moulstil relaterDingsbums1.</p>';
 $Regal[] = '</div>';

 $Regal = implode('', $Regal);

 echo $Regal;
}
```

Und zu guter letzt der Stil mit weiteren Parametern.

Einiges kann man viel effizienter coden. Ich bleib absichtlich bei den if-Schleifen und spare mir einige Prüfungen. Frag einfach nach, bei "komischen" Ausgaben, unerwarteten Fehlern.

```
function modChrome_relaterDingsbums1($module, &$params, &$attribs)
{
 $Regal = array();

 $moduleClass = $params->get('moduleclass_sfx', '');
 $moduleTag = $params->get('module_tag', 'div');
 $bootstrapClass = $params->get('bootstrap_size', 0);
 $headerTag = $params->get('header_tag', 'h1');
 $headerClass = $params->get('header_class', 'meineEigeneKlasse');

 // Nur, falls nicht 0 (sonst ergäbe das Klasse span0),
 if ($bootstrapClass)
 {
  // dann hänge diese Bootstrap-CSS-Klasse an die $moduleClass dran.
  $moduleClass .= ' span' . $bootstrapClass;
 }

 $Regal[] = '<' . $moduleTag;
 if ($moduleClass)
 {
  $Regal[] = ' class="' . $moduleClass . '"';
 }
 $Regal[] = '>';

 if ($module->showtitle)
 {
  $Regal[] = '<' . $headerTag . ' class="' . $headerClass . '">';
  $Regal[] = $module->title;
  $Regal[] = '</' . $headerTag . '>';
 }

 $Regal[] = $module->content;
 $Regal[] = '<p class="small">Ich bin Moulstil relaterDingsbums1.</p>';
 $Regal[] = '</' . $moduleTag . '>';

 $Regal = implode('', $Regal);

 echo $Regal;
}
```

Und das ergibt mit diesen Einstellungen:



im Seitenquelltext (gehübscht):

```
<section class="ichBinKlasse weitereKlasse span6">
 <h2 class="ichBinHeaderKlasse undNochEine">Mein Testmodul</h2>
 <div class="customichBinKlasse weitereKlasse"  >
  <p>Hallo Wolken. Hallo dies das.</p>
  <p>Wo geht's hin?</p>
 </div>
 <p class="small">Ich bin Moulstil relaterDingsbums1.</p>
</section>
```

Dass `weitereKlasse` doppelt vorkommt, liegt daran, dass auch das Modul in seiner Ausgabedatei `/modules/mod_custom/tmpl/default.php` den `$moduleclass_sfx` drin hat. Könnte man bspw. durch einen Modul-Override dort entfernen.

Da geht selbstverständlich noch viel mehr mit Modulstilen. Module je nach Seite "einfärben", nach Kategorie, sonstwas zusätzlich ausgeben usw. usf.