

Ein kommentierter Aufbau eines Templates für Mambo 4.5.1 stable

Im wesentlichen baut dieses Tutorial auf das für die Mamboversion 4.5.x vom gleichen Autor auf. Es wurde überarbeitet und für die Mamboversion 4.5.1 stable angepaßt.

Erweitert wird dieses Tutorial mit einer Erklärung der *Page_Class_Suffix*, die innerhalb verschiedener Parametern angegeben werden können.

In diesem Tutorial wurde desweiteren wieder die CSS-Dokumentation von Christian Hent eingefügt.

Dieses Template darf unter Nennung des Copyrights frei weitergegeben werden.

Für die Richtigkeit der Angaben wird keine Gewähr übernommen.

Die Autoren wünschen viel Spaß und Erfolg beim Template bauen!

Axel Tüting
www.mamboportal.ch
www.time4mambo.de
tueting@time4mambo.de

Christian Hent
www.mamboreport.de
c.hent@nefkom.net

Inhalt

Ein kommentierter Aufbau eines Templates für Mambo 4.5.1 stable.....	1
Aufbau des Templates – Kopf:.....	3
Publikation des eigenen Templates	7
Uploaden	9
Überblick über die benötigten Dateien:.....	9
Menüpunkt <i>Module Positions</i>	10
Class Suffix.....	10
CSS-Datei.....	11
CSS-Definitionen.....	12

Aufbau des Templates – Kopf:

```
<?php
```

Beginn des PHP-Codes – Wichtig: Davor darf nichts stehen. Auch keine Leerzeile/-zeichen

```
defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
```

Falls unberechtigte Zugriffe in diesem Verzeichnis stattfinden → Fehlerabfangeroutine

```
// needed to separate the ISO number from the language file constant _ISO  
$iso = split( '=', _ISO );
```

Was hier genau passiert, weiß ich leider auch nicht.

```
// xml prolog  
echo '<?xml version="1.0" encoding="'. $iso[1] ."'? .>';  
?>
```

Vorbereitung für die XML-Angaben

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">
```

Hiermit beginnt der eigentliche HTML/XHTML-Teil

```
<head>  
<?php  
if ( $my->id ) {  
    initEditor();  
}  
?>
```

Damit öffnet sich im admin-Bereich der Editor

```
<meta http-equiv="Content-Type" content="text/html; <?php echo _ISO; ?>" />  
<?php mosShowHead(); ?>
```

Zur Laufzeit werden hier die Sprach- und Zeichennutzungscode eingefügt. Wichtig für den Browser, damit er weiß, mit welchem Schriftcode er sich auseinandersetzen muß. (Zu den obigen ISO-Angaben kann ich leider nichts sagen.)

```
<link rel="shortcut icon" href="<?php echo $mosConfig_live_site;?>/images/favicon.ico" />
```

Laden des Icons, welches im Favoriten des Browsers angezeigt wird, wenn man die Seite bookmarkt.

```
<link rel="stylesheet" type="text/css" href="<?php echo $mosConfig_live_site;  
?>/templates/JavaBean/css/template_css.css" />
```

Laden der CSS-Datei – der Pfad muß bei eigenen Templates angepaßt werden.

```
</head>
```

<body>

Das eigentliche Aussehen des Templates steht an dieser Stelle zwischen den BODY-Tags

</body>

</html>

Nachfolgend wird eine einfache Tabelle erstellt, die zwischen den BODY-Tags eingefügt wird.
Bei Problemen mit HTML empfehle ich <http://de.selfhtml.org/>
Die wirklich ausgezeichnete HTML- und CSS-Dokumentation kann dort kostenlos downgeloadet werden.

Zur Erklärung eine einfache Tabelle mit folgendem Aussehen:

Kopf – Zeile 1 über 3 Spalten		
Linke Seite oben Zeile 2, Spalte 1	Mitte oben Zeile 2, Spalte 2	Rechte Seite oben Zeile 2, Spalte 3
Linke Seite unten Zeile 3, Spalte 1	Mitte unten Zeile 3, Spalte 2	Rechte Seite unten Zeile 3, Spalte 3

```
<table>
  <tr>
    <td colspan=3>
      <!-- Zeile 1 über drei Spalten -->
    </td>
  </tr>
  <tr>
    <td>
      <!--Zeile 2 Spalte 1-->
    </td>
    <td>
      <!--Zeile 2 Spalte 2-->
    </td>
    <td>
      <!--Zeile 2 Spalte 3-->
    </td>
  </tr>
```

```

<tr>
  <td>
    <!--Zeile 3 Spalte 1-->
  </td>
  <td>
    <!--Zeile 3 Spalte 2-->
  </td>
  <td>
    <!--Zeile 3 Spalte 3-->
  </td>
</tr>
</table>

```

Um Inhalte in die einzelnen Tabellenzellen zu bekommen, bediene ich mich der nachfolgenden PHP-Mambo-Elemente:

Komponente Banner einfügen

```
<?php mosLoadComponent( "banners" ); ?>
```

Den Seitennamen einfügen

```
<?php echo $mosConfig_sitename; ?>
```

Den aktuellen Pfad anzeigen lassen

```
<?php include "pathway.php";?>
```

Alle Module die im Bereich „Top“ angezeigt werden sollen

```
<?php mosLoadModules('top'); ?>
```

Alle Module die im Bereich „left“ angezeigt werden sollen

```
<?php mosLoadModules('left'); ?>
```

Alle Module die im Bereich „right“ angezeigt werden sollen

```
<?php mosLoadModules('right'); ?>
```

Alle Module die im Bereich „user1“ angezeigt werden sollen

```
<?php mosLoadModules('user1'); ?>
```

Alle Module die im Bereich „user2“ angezeigt werden sollen

```
<?php mosLoadModules('user2'); ?>
```

Alle Module die im Bereich „bottom“ angezeigt werden sollen

```
<?php mosLoadModules('bottom'); ?>
```

Alle Module die im Bereich „insert“ angezeigt werden sollen

```
<?php mosLoadModules('insert'); ?>
```

Der eigentliche Inhalt

```
<?php include_once("mainbody.php"); ?>
```

Seit der *Mambo 4.5.1 stable* gibt es eine Unzahl weiterer Positionen, die ich in meinem Templates einbauen kann. Welche das sind, kann man/frau nachschauen, wenn man/frau in das Backend (also in den Adminbereich) geht, dort den Modulmanager aufruft, ein beliebiges Modul editiert und unter **Positionen** nachschaut. Und genau diese Positionen spiegeln die obigen Angaben wider.

Also wird mit **Top, left, right, etc.** bei der jeweiligen Modul-/Komponentenanzeige im Backend eingestellt, an welcher Stelle das Modul erscheinen soll.

Und bei uns im Template wird die Stelle, die Position, entsprechend definiert.
Die jeweiligen Mamboelemente werden dabei einfach zwischen den TD-Tags geschrieben:

```
<td><?php mosLoadModules('left'); ?> </td>
```

In Zukunft werden jetzt alle Module, die links (left) erscheinen sollen, an dieser Stelle im Template ausgegeben.

Auf diese Weise kann ich meinem Template feste Bereiche zuordnen. Beispielsweise:

```
<table>
  <tr>
    <td colspan=3>
      <!-- Zeile 1 über drei Spalten -->
      <?php mosLoadModules('top'); ?>
    </td>
  </tr>
  <tr>
    <td>
      <!--Zeile 2 Spalte 1-->
      <?php mosLoadModules('left'); ?>
    </td>
    <td>
      <!--Zeile 2 Spalte 2-->
      <?php include_once("mainbody.php"); ?>
    </td>
    <td>
      <!--Zeile 2 Spalte 3-->
      <?php mosLoadModules('right'); ?>
    </td>
  </tr>
  <tr>
    <td>
      <!--Zeile 3 Spalte 1-->
      <?php mosLoadModules('user1'); ?>
    </td>
    <td>
      <!--Zeile 3 Spalte 2-->
      <?php mosLoadComponent( "banners" ); ?>
    </td>
    <td>
      <!--Zeile 3 Spalte 3-->
      <?php mosLoadModules('bottom'); ?>
    </td>
  </tr>
</table>
```

Auf diese Weise lassen sich viele unterschiedliche Aufbauten von Templates für Mambo entwickeln. Module und Menüs können an jeder beliebigen Stelle im Template ausgegeben werden. Selbstverständlich muß dafür nicht zwingenderweise eine Tabelle genommen werden, jeder HTML-Tag kann dafür benutzt werden.

Die Farbgestaltung, etc. findet schwerpunktmäßig über CSS statt. Wobei natürlich auch die HTML-Attribute im BODY-, TABLE- oder anderen HTML-Tags benutzt werden können. Aber dann sind die Templates nicht mehr so flexibel. Denn wie wir weiter unten sehen können, gibt es weitere Möglichkeiten, ein Template zu beeinflussen. Und das geht am dynamischsten, wenn das eigentliche Template nur das Nötigste enthält.

Für CSS zwei Möglichkeiten:

- Direkt unter `<td style="..." >` eingeben.
- Oder die CSS-Datei, die sich standardmäßig im jeweiligen Templateverzeichnis unter „css“ befindet und `template_css.css` heißt verändern, erweitern und anpassen.

Publikation des eigenen Templates

Zum Publizieren des eigenen Templates ist zunächst die XML-Datei sehr wichtig: Diese muß `templateDetails.xml` heißen.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<mosinstall type="template">
```

Was soll installiert werden?

```
<name>EigenerTemplateName</name>
```

Unter welchem Namen wird das Template installiert

```
<creationDate>21. Oktober 2004</creationDate>
```

Installationsdatum

```
<author>Axel Tueting</author>
```

Autor des Templates (deutsche Umlaute möglichst vermeiden), also vermutlich Dein Name

```
<copyright>Dieses Template unterliegt der GNU/GPL Lizenz</copyright>
```

Wenn es ein freies Template ist, dann kann es hier entsprechend genannt werden, in dem auf die GNU/GPL Lizenz verwiesen wird, die einiges zum Thema Freeware und OpenSource festgeregelt hat.

Ist es kein freies Template kann folgendes eingefügt werden:

Dieses Template unterliegt **nicht** der GNU/GPL Lizenz

Oder:

Dieses Template wurde für die Firma xyz erstellt und unterliegt deren Copyright

`<authorEmail>tueting@time4mambo.de</authorEmail>`

Die eMail-Adresse des Autors – also vermutlich Deine eigene

`<authorUrl>www.time4mambo.de</authorUrl>`

URL der Seite des Autors.

`<version>1.0</version>`

Version des Templates – besonders interessant, wenn das Template weiterentwickelt wird

`<description>Dieses ist ein Beispiel-Template</description>`

Kurze Beschreibung des Templates

`<files>
<filename>index.php</filename>`

Das Template-File selber. In aller Regel ist das die “index.php”, welches dann am Anfang auch geladen wird

`<filename>template_thumbnail.png</filename>`

Ein verkleinerter Screenshot des Templates. Wichtig bei der Auswahl im Backend und beim Templatechooser.

`</files>
<images>
<filename>images/hintergrundgrafik.jpg</filename>`

Wenn Grafiken im Template benutzt werden (beispielsweise Hintergrundbilder), dann müssen diese mit korrekten (relativen) Pfad angegeben werden.

Der standardmäßige Verzeichnisbau bei den Templates:

- templates
 - EigenerTemplateNameVerzeichnis
 - css
 - images

`<filename>images/hintergrundgrafik2.jpg</filename>`

Werden mehrere Bilder benutzt, muß jede Grafik in einem filename-Tag stehen.

`</images>`

`<css>
<filename>css/template_css.css</filename>`

Der Name der zugehörigen CSS-Datei mit korrekten (relativen) Pfad

`</css>
</mosinstall>`

Auf einen Blick:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<mosinstall type="template">
  <name> EigenerTemplateName </name>
  <creationDate>15. August 2004</creationDate>
  <author>Axel Tueting</author>
  <copyright>Dieses Template unterliegt nicht der GNU/GPL Lizenz</copyright>
  <authorEmail>tueting@ time4mambo.de</authorEmail>
  <authorUrl>www. time4mambo.de</authorUrl>
  <version> 1.0</version>
  <description>Dieses ist ein Beispiel-Template</description>
  <files>
    <filename>index.php</filename>
  </files>
  <filename>template_thumbnail.png</filename>
  </files>
  <images>
    <filename>images/ hintergrundgrafik.jpg</filename>
    <filename>images/ hintergrundgrafik2.jpg</filename>
  </images>
  <css>
    <filename>css/template_css.css</filename>
  </css>
</mosinstall>
```

Uploaden

Es gibt zwar auch die Möglichkeit, direkt über das Backend und dem Menü *Site/Template Manager/Install* das Template hochzuladen, aber es geht auch "per Hand" – also die Dateien in ihrer Ordnerstruktur einfach in das *Template*-Verzeichnis laden.

Ich selber habe bislang nicht so gute Erfahrungen mit der Install-Routine im Backend gemacht.

Überblick über die benötigten Dateien:

- index.php
Dort ist das eigentliche Template enthalten. In unserem Beispiel also die einfache Tabelle mit den Bereichszuordnungen.
- templateDetails.xml
Die Installationsanweisungen
- template_css.css
Mit den CSS-Angaben für das eigentliche Aussehen des Templates
- Grafiken im JPG, GIF oder PNG-Format für Hintergründe oder fest vorgegebene Grafiken im Template, bzw. des Screenshots

In Deinem Mambo jetzt im Templateverzeichnis einen neuen Ordner anlegen mit „MeinTemplateVerzeichnisNamen“ und die Dateien hochladen.

Im Backend nun in das Menü *Site/Template Manager* gehen und Dein Template auswählen und es *publishen*

Menüpunkt *Module Positions*

Site/Template Manager/Module Positions

Um nicht den Überblick über seine vielen Positionen zu verlieren, lassen sich hier mittels einer Kurzbeschreibung festhalten, wofür die einzelnen Positionen gedacht sind. Also handelt es sich hier quasi um eine "Merkliste".

Class Suffix

Module Class Suffix für die Module

Page Class Suffix für die Menüs und Artikel

Damit kann ich zusätzlichen Einfluß auf die Template nehmen. Wenn ich eine Suffix angebe und diese in der CSS-Datei übernehme, kann ich damit das Aussehen des Moduls, Menüs, Artikels beeinflussen.

Beispiel:

Page Class Suffix: *MeinFormat*

Nun die CSS-Datei editieren und folgendes eingeben:

```
.MeinFormat
{
...meine Format-Angaben ...
}
```

Der Artikel oder das Menü wird jetzt so ausgegeben, wie ich es in *MeinFormat* definiert habe.

Damit ergeben sich interessante dynamische Effekt-Möglichkeiten, bzw. unterschiedlich aussehende Seiten bei bestimmten Artikeln und vielem mehr.

CSS-Datei

Die Template-CSS-Datei lässt sich ganz den eigenen Bedürfnissen und Vorstellungen anpassen. Jeder HTML-Tag kann mit eigenen Formaten belegt werden, eigene Bereiche und Klassen lassen sich definieren und natürlich auch verschachtelte CSS-Notationen. Damit sind Mambo vom designerischen nur wenige Grenzen gesetzt.

Zu beachten gilt aber, dass einzelne Komponenten und Module, sowie Textbausteine, bestimmte CSS-Definitionen erwarten. Sie stehen fertig programmiert im Quellcode drin.

Deshalb sollten bestimmte CSS-Klassen unbedingt übernommen werden. Diese können aber dann natürlich nach eigenen Vorstellungen angepaßt werden.

Mambo Open Source (Template) CSS Definitionen

Einleitung

Cascading Style Sheets (CSS) ist eine Sprache für strukturierte Dokumente wie beispielsweise HTML und vereinfacht die Pflege aufwendiger Webseiten durch die Trennung von Layout und Inhalt. CSS greift auf Elemente von HTML zu und verleiht diesen individuelle Merkmale. Mittels CSS werden Schriften, Farben, Hintergründe, Tabellen, Grafiken, Rahmen, Abstände (innen und aussen) etc. formatiert, ohne dass dabei die entsprechende Definition jedesmal neu angegeben werden muss. Formatierungsanweisungen werden in einer speziellen Datei mit der Kennung ".css" (**template_css.css**) abgelegt und mittels LINK Element mit den eigentlichen Inhalten (**index.php**) verknüpft.

Übersicht: Aufbau des Mambo Templates



Einbinden der CSS Datei

```
<link href="<?php echo
$mosConfig_live_site;?>/templates/dein_template/css/template_css.cs
s" rel="stylesheet" type="text/css" />
```

Beispieldefinition aus der CSS Datei

```
table.moduletable th {
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-size: 12px;
    color: Gray;
    text-align: left;
    width: 100%;
    height: 18px;
    border-bottom: 1px solid #CCCCCC;
    background: transparent;
```

Die Formatdefinitionen werden in Klassen zusammengefasst. Diese werden durch einen Punkt und dem Klassennamen definiert – **table.moduletable th** formatiert z. B. den Tabellenkopf einer zuvor festgelegten Tabelle.

Ich habe versucht eine (nahezu) vollständige Liste aller in Mambo OS 4.5 Templates zu nutzenden CSS Klassen zu erstellen und stelle sie hiermit frei zur Verfügung.

	Name	Verwendung/Definition	Beschreibung
Mambo General Styles – Allgemeine CSS Definitionen			
01	h1 - h6	<h4>Überschrift Grösse vier</h4>	formatiert Überschriften in einem Dokument
02	tr	<tr>Tabellenzeile</tr>	formatiert eine Zeile in einer Tabelle
03	th	<th>Kopfzelle</th>	formatiert den Kopf einer Tabelle
04	td	<td>Tabellendaten</td>	formatiert Tabellenzelle als Behälter für Daten, Text oder Bilder
05	p	<p>Absatz</p>	formatiert einen beliebigen Absatz
06	div	<div><ID=myMenuID></div>	formatiert einen Container für HTML-Elemente, denen Stylesheet-Eigenschaften zugewiesen werden
07	.sitetitle		formatiert den Titel der Seite
08	.searchbox	<input class="searchbox">	formatiert das Suchfeld der Seite
09	.small		formatiert diverse Einträge, u.a. den Namen des Autors - '...geschrieben von'
10	.pagenav		formatiert die '<< Start < Previous 1 Next > End >>' Links
Mambo Form Styles - Formulardefinitionen			
11	.inputbox	<input class="inputbox" size="10" />	formatiert ein Texteingabefeld
12	.button	<input type="submit" class="button"/>	formatiert die Schaltfläche 'senden'
Mambo Tabbed Frontend Admin Interface – Definition des Content Edit Modus (Frontend)			
13	.ontab	<td class="ontab">Images</td>	formatiert aktiven Karteireiter des Editors beim Editieren im Frontend
14	.offtab	<td class="offtab">Images</td>	formatiert nicht aktiven Karteireiter des Editors beim Editieren im Frontend
15	.tabpadding	<td class="tabpadding"> </td>	formatiert die Grösse der Karteireiter des Editors beim Editieren im Frontend
16	.pagetext	<div id="page1" class="pagetext">	formatiert geöffneten Editor HtmlArea2 ?
Mambo Menu Styling – Definition der Menus (Hauptebene bzw. Unterebene)			
17	.mainlevel	.mainlevel:link .mainlevel:hover .mainmenu:active .mainlevel:visited	Link im normalen Zustand, Hauptmenu Link bei MouseOver Link gerade aktiv Link bereits besucht
18	.sublevel	.sublevel:link .sublevel:hover .mainmenu:active .sublevel:visited	Link im normalen Zustand , Untermenu Link bei MouseOver Link gerade aktiv Link bereits besucht

Beispieldefinition für Menus

```

a.mainmenu:link, a.mainmenu:visited {
    color: #4D6165;
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-weight: normal;
}

a.mainmenu:hover, {
    color: #B0C4DE;
    text-decoration: underline;
}

a.sublevel:link, a.sublevel:visited {
    color: #4D6165;
    font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
    font-weight: normal;
}

```

	Name	Verwendung/Definition	Beschreibung
Mambo General styling for Sect. - Categ. - Items – Definition der Rubriken- Kategorien und Inhalte			
19	.createdate		formatiert das Erstellungsdatum des Artikels
20	.modifydate	<td class="modifydate" >	formatiert das Veränderungsdatum des Artikels
21	.readon		formatiert den ...Weiterlesen Link im Artikel
Mambo Styles for Sections – Definition der Rubriken			
22	.sectiontableheader	<td class="sectiontableheader">	formatiert Tabellenkopf in Rubriken
23	.sectiontableentry1	<tr class="sectiontableentry1">	formatiert Einträge in Rubriken
24	.sectiontableentry2	<tr class="sectiontableentry2">	formatiert Einträge in Rubriken
Mambo Styles for Categories – Definition der Kategorien			
25	.category	<td class="category">	
Mambo Styles for Item – Definition der Inhalte			
26	.contentpane	<table class="contentpane">	Formatierung wird des öfteren verwendet, u.a. in dem Bereich News (Tabelle mit Rubriken und der Beschreibung + Grafik)
27	.contentheading	<td class="contentheading">	formatiert die Überschrift eines Artikels
28	.contentdescription	<td class="contentdescription">	formatiert die Beschreibungen der News, Weblinks, etc...
29	.content_rating		formatiert die Anzeige der bereits vollzogenen Abstimmungen ...User Rating: x/x
30	.content_vote		formatiert die Anzeige der vorzunehmenden Abstimmung ...Poor/Best
Mambo Styles for components – Definitionen der Komponenten			
31	.componentheading		formatiert die Anzeige des Titels einer Komponente im Frontend der Website
Mambo Styles for modules – Definitionen der Module			
32	.moduletable	<table class="moduletable">	formatiert die Tabelle zur Anzeige der Module
33	.moduletable th	<th class="moduletable">	formatiert den Kopf der jeweiligen Tabelle
34	.moduletable td	<td class="moduletable">	formatiert die Zellen der Tabelle
Mambo's Built-in Component's Style – Definition der integrierten Komponenten			
35	.weblinks	<td class="contentdescription">	formatiert die Einträge in den Rubriken der Weblink Komponente
36	.newsfeedheading		formatiert den Namen des eingebundenen Feeds jedoch nicht dessen Titel (Überschriften)
37	.newsfeeddate		formatiert die Anzeige des Datums im Newsfeedsmodul
38	.fase4rdf	<td class="fase4rdf">	formatiert den anzuzeigenden Inhalt des Feeds
39	.poll	<table class="moduletable">	formatiert die Text Anzeige im Umfrage Modul
40	.pollstableborder	<table class="moduletable">	formatiert die Rahmen des Umfragemoduls
41	searchintro	<table class="searchintro">	formatiert die Antwort einer Suchanfrage ... Insgesamt x Ergebnisse. Suche nach x mit ...
42	.contact	<table class="contact">	formatiert die 'Contact us' Tabelle, jedoch nicht das Dropdown zur Auswahl der Kontaktperson
43	.contact td.icons	<table class="contact">	formatiert die Zelle der Tabelle welche das Icon zum Kontakt beinhaltet
44	.contact td.details	<td class="details">	formatiert die Details zum jeweiligen Kontakt wie Adressen, Telefonnummern, etc...

Ergänzungen und Korrektur			
1	<code>.contentpaneopen</code>	<code><td class="contentpaneopen"></code>	formatiert den Inhalt eines Artikels
2	<code>.newsflash</code>	<code><td class="newsfeedheading"></code>	formatiert den Inhalt aus der Newsflash Komponente

Bitte fehlende Definitionen gegebenenfalls an c.hent@nefkom.net senden. Danke