

Inhaltsverzeichnis

Inhaltsverzeichnis.....	1
Über dieses Dokument.....	1
Aufbau des Sitefiles	2
Aufbau eines Templates.....	3
Aufbau eines Contentfiles	4
Kleine Referenz für die Erstellung von Contentfiles	5
Aufbau einer Section.....	6

Über dieses Dokument

Erstellt in Zusammenarbeit mit Andrej Bisin von www.hardcoding.net. Eine aktuelle Version dieses Dokumentes können sie unter www.masoweb.net unter „Sonstiges“ herunterladen.

Das Dokument ist kostenlos und darf beliebig oft weiterverteilt werden, solange der Absatz „Über dieses Dokument“ bestehen bleibt.

© 2002 Martin Sommer (ma.so@gmx.net)

Aufbau des Sitefiles

Der Header `<?xml version="1.0" ?>` und der Tag `site` sind der Start eines jeden Site Files, der Hauptpfad (`siteroot`) und der Seitentitel (`title`) der Homepage werden unter `site` angegeben.

```
<?xml version="1.0" ?>
<site title="MaSoWeb" siteroot="">
```

Templates sind Vorlagen, die später das Aussehen der Seite bestimmen, man kann beliebig viele Templates benutzen, aber jeweils nur eins pro Seite. `printversion` gibt ein `template` an, das möglichst einfachen HTML Code ohne Tabellen und nur wichtige Elemente beinhaltet. `id` und `file` müssen immer angegeben werden.

```
<templates>
  <template id="standard" file="templates/standard.html" printversion="templates/printversion.html"/>
</templates>
```

`resources` dienen zur Vordefinition von Quellen, bei denen es sich nicht um Seiten handelt. Sie enthalten immer die Attribute `url` und `id`. Die Angabe des Attributs `name` ist sinnvoll, so können z.B. Downloads dynamisch gelinkt werden. Wenn im Contentfile `` angegeben wird, kommt in der generierten Site automatisch der Name als Link dazu. Ist dies nicht gewünscht muss in klassischer Art `Linkname` geschrieben werden. Bilder können in den Contentfiles mit dem Befehl `` geladen werden. Auch Javascripts und externe Links sind möglich. Der Tag `type` sollte mit angegeben werden. So lassen sich die unterschiedlichen Ressourcen auch durch Programme unterscheiden.

```
<resources>
  <resource id="einstein-small" name="Einsteinbild klein" url="images/einstein-small.jpg" type="image/jpeg"/>
  <resource id="dl_mailit" name="mailit.zip (4,5kB)" url="download/mailit.zip" type="download/zip"/>
  <resource id="einstein-zoom" name="Zoom Bild" url="javascript:zoomImage('einsteinbig.jpg')" type="javascript"/>
</resources>
```

Unter dem Tage `pages` werden sämtliche Seiten notiert, die es später auf der Homepage geben soll, jede Seite sollte nach folgendem Schema angegeben `<page`

```
as="verzeichnis/dateiname.html"
```

`keywords="keyword1, keyword2"` `template="templateid"/>`, dick gedruckte Attribute sind zwingend. `file` wird immer aus Sicht des `siteroot`- Pfades angegeben. Die beiden Attribute `id` und `name` sind für die spätere Verlinkung um Contentfile nötig. Wird dort `` geschrieben, so erscheint auf der generierten Seite ein Link zur `pageid` passenden Seite mit dem im Sitefile angegebenen Namen. Soll statt dessen ein anderer Name erscheinen so muss der Link auf klassische HTML Weise `Name des Links` notiert werden. `as` dient stellt die Möglichkeit dar, das gleiche Contentfile unter zwei verschiedenen Adressen darzustellen, unter `as` wird der Dateiname angegeben, den die Datei „scheinbar“ haben soll. `template` und `keywords` werden auf darunterliegende Seiten vererbt, `template` muss also nur in der hierarchisch höheren Seite nötig, kann aber auch tiefer angegeben werden, wenn ein anderes Template gewünscht ist. `keywords` muss nicht angegeben werden, sollte aber, die enthalten Keywords werden später in den META TAG KEYWORDS mit hinein generiert. Wenn Keywords gewünscht sind, die nicht auf darunterliegende Seiten vererbt werden sollen, können diese auch im Contentfile angegeben werden. Wichtig beim Erstellen der Struktur ist das Beachten der Tagenden mit dem `/`. Hat eine Seite noch Unterseiten, so wird am Ende des Tags kein `/` gemacht; sobald alle Unterseiten eingetippt sind muss der Tag mit `</page>` geschlossen werden. Der `/` am Ende stellt also nur ein Abkürzung dar, wenn es keine Unterseiten gibt.

```
<pages>
  <page name="Home" file="index.html" id="home" template="standard" keywords="" />
  <page name="Software" id="software" file="software/index.html" keywords="software, script, scripts, perl, cgi, javascript">
    <page name="Mail It" id="mailit0" file="software/mailit0.html" keywords="mail, perl, script, id, liste, popup, cool, mail_it, mail, it">
      <page name="Genaue Beschreibung" id="mailit1" file="software/mailit1.html" keywords="beschreibung, genaue"/>
      <page name="Installation und Konfiguration" id="mailit2" file="software/mailit2.html" keywords="konfiguration, installation"/>
      <page name="Demo" id="mailit3" file="software/mailit3.html" keywords="demo"/>
    </page>
  </page>
</pages>
```

Wie alle anderen Tags auch müssen `pages` und `site` geschlossen werden.

```
</pages>
</site>
```

Aufbau eines Templates

Das Template (Vorlage) ist eine normale HTML Datei, in der zusätzlich ein paar spezielle Tags verwendet werden, die später vom CMS mit den entsprechenden Inhalten versehen werden. Das Beispiel soll klar machen, was damit gemeint ist und wie es angewendet wird.

```
<!-- INCLUDE STYLESHEET style.xss --> <!-- lädt das Stylesheet style.xss (siehe Aufbau einer Section -->
<html>
<head>
  <title>{sitetitle} - {title}</title>
  <meta name="keywords" content="{keywords}" />
  <meta name="description" content="{description}" />
  <meta name="date" content="{meta_date}" />
  <link rel="stylesheet" href="{siteroot}/style/style.css" type="text/css" />
</head>

<body>
<p>
<a href="{siteroot}/index.html">Home</a> |
<a href="{last_url}">Zurück</a> |
<a href="{next_url}">Weiter</a> |
<a href="{this_url}?show=printerversion">Druckversion</a>
</p>
<p>
{hmenu}
</p>
<h1>{title}</h1>
<!-- BLOCK START section --> <!-- Sectionbeginn, alles ab hier wird so oft wiederholt, wie es Sections in der Content Datei gibt -->
<p>
<h3>{secttitle}</h3><br />
<br />
{sectcontent}
</p>
<!-- BLOCK END section --> <!-- Sectionende, hier ist der zu wiederholende zu Ende -->
<p>
Letzte Änderung: {content_modtime}
</body>
</html>
```

In geschwungenen Klammern angegebene Begriffe stehen immer für eine bestimmte Funktion, entweder für eine ins CMS integrierte Funktion oder für eine Extension. Die folgende Tabelle gibt Auskunft über alle möglichen Begriffe, die fest ins CMS integriert sind.

Begriff	Bedeutung
{sitetitle}	Titel der Seite, wird aus dem Sitefile entnommen
{title}	Titel der aktuell geladenen Seite, wird standardmäßig dem Sitefile entnommen, wenn in Contentfile kein anderer angegeben wurde
{keywords}	Keywords, werden aus dem Sitefile entnommen
{description}	Beschreibung der Seite, wird dem Contentfile entnommen
{meta_date}	Meta Datum, wird anhand des Dateidatums in der Form JJJJ-MM-TT ausgegeben
{content_modtime}	Änderungsdatum, wird anhand des Dateidatums in der Form TT.MM.JJJJ ausgegeben
{secttitle}	Section Titel, wird der jeweiligen Section im Contentfile entnommen
{sectcontent}	Section Content, wird der jeweiligen Section im Contentfile entnommen
{siteroot}	Quellpfad der Seite, wird dem Sitefile entnommen
{last_url}	nächste Page, wird anhand der Seitenstruktur im Sitefile ermittelt
{next_url}	vorherige Page, wird anhand der Seitenstruktur im Sitefile ermittelt
{this_url}	eigene URL
{parent_url}	darüber liegende Seite (laut Sitefile)

Es werden einige Extensions für das Template bereits mitgeliefert

Extension/Aufruf	Beschreibung
index.php/{index}	
sitemap.php/{sitemap}	Eine hierarchische Sitemap
vmenu.php/{ExtVMenu}	Vertikales Menu
hmenu.php/{ExtHMenu}	Horizontales Menu
location.php/{location}	

Weitere Informationen zu Extensions auch in der Datei „Mitgelieferte Extensions.doc“.

Aufbau eines Contentfiles

Contentfiles sind alle Dateien, die unter `pages` im Sitefile angegeben sind. Sie beginnen mit dem Header `<?xml version="1.0" ?>` und dem Tag `<contentpage>`. In `contentpage` können die Parameter `type` und `keywords` benutzt werden. Standardmäßig wird `type="xhtml"` verwendet, was in den meisten Fällen korrekt ist. Es braucht nicht extra angegeben werden. Unter `keywords` können Keywords notiert werden, die nicht auf darunter liegende Seiten (siehe Sitefile) vererbt werden sollen.

```
<?xml version="1.0" ?>
<contentpage type="xhtml" keywords="allgemeine, beschreibung" >
```

`description` ist die Beschreibung einer Seite, sie wird später unter dem META TAG DESCRIPTION zu finden sein. `title` ist ebenso optional und bietet die Möglichkeit einen genaueren Titel für die Seite anzugeben. Wenn `title` nicht eingebaut ist, wird als Titel im generierten File der im Sitefile verwendete benutzt.

```
<description>Image Zoomer öffnet ein neues Fenster für die große Ansicht eines kleines Bildes, dass passgenau zum Bild ist.</description>
<title>Image Zoomer Allgemeine Beschreibung</title>
```

`content` startet den Inhaltsbereich der jeweiligen Seite.

```
<content>
```

Der Content-Bereich ist in Sections (Abschnitte) aufgeteilt. Jeder Abschnitt kann einen Titel bekommen, der mit dem Attribut `title` festgelegt wird. Die Sections werden bei der Generierung hintereinander abgearbeitet und stehen dann auch in der entsprechenden Reihenfolge. Wo der Sectiontitel und der dazugehörige Inhalt befindet, wird über das Template geregelt. Was in den Sections alles möglich ist, ist auf auf der Seite „Aufbau einer Section“ erklärt.

```
<sect title="Was ist Image Zoomer?">
Image Zoomer öffnet ein neues Fenster für die große Ansicht eines kleines Bildes, dass passgenau zum Bild ist.
</sect>
<sect title="Mehr zu Image Zoomer">
<ext name="masoweb_index" />
</sect>
<sect title="Download">
<a ref="dl_imagezoomer"/>
</sect>
```

Wie alle Tags müssen auch `content` und `contentpage` am Ende geschlossen werden

```
</content>
</contentpage>
```

Kleine Referenz für die Erstellung von Contentfiles

fett gedrucktes bezeichnet Angaben, die auf jeden Fall gemacht werden sollten bzw. müssen

kursiv gedrucktes bezeichnet Angaben, die individuell angepasst werden müssen

normal gedrucktes bezeichnet optionale Parameter, die gemacht werden können, aber nicht müssen

Tag	Bedeutung / Wirkung
<code>
</code>	Erzeugt einen Zeilenumbruch
<code></code>	Erzeugt einen Link mit dem Namen aus dem Sitefile
<code>Linkname</code>	Erzeugt einen Link mit dem Namen "Linkname"
<code></code>	Erzeugt den Pfad zum Image automatisch
<code><table class="table_klasse" cellpadding="0" cellspacing="0" border="0">Tabellentags</table></code>	Erzeugt eine Tabelle
<code><tr>Tabellenzellen</tr></code>	Erzeugt eine Tabellenzeile
<code><td class="td_klasse" align="center" valign="top">Zelleninhalte</td></code>	Erzeugt eine Tabellenzelle
<code><sect title="Titel">Sectioninhalt</sect></code>	Erzeugt eine Section
<code><title>Sondertitel einer Seite</title></code>	Setzt für die Seite einen Extratitel, der von dem im Sitefile angegebenen abweicht
<code><contentpage keywords="viele, Keywords">Inhalt</contentpage></code>	Zusätzliche Keywords, die nicht auf darunterliegende Seiten vererbt werden sollen
<code><nobr>Wörter ohne Umbruch</nobr></code>	Erzielt, dass kein Umbruch gemacht wird

Sonderzeichen:

 Festes Freizeichen
 & & (Kaufmännisches UND)
 < < (kleiner als/lower than)
 > > (größer als/greater than)

Aufbau einer Section

Die Sections sind die Abschnitte, in denen sich der wirkliche Inhalt befindet. Sie beginnen immer mit dem Tag `<sect>`. Es kann der Parameter `title` verwendet werden, mit dem der Sectiontitel angegeben wird.

```
<sect title="Was ist Image Zoomer?">
Image Zoomer öffnet ein neues Fenster für die große Ansicht eines kleinen Bildes, das passgenau zum Bild ist.
</sect>
<sect title="Mehr zu Image Zoomer">
<ext name="masoweb_index"/>
</sect>
```

In den Sections können verschiedene zusätzlich zum normalen HTML-Code verfügbare Tags verwendet werden, die mit der folgenden Tabelle verdeutlicht werden sollen:

Tag	Nutzen
<code></code>	Erzeugt automatisch einen Link mit dem Namen, der im Sitefile angegeben
<code>Linkname</code>	„Klassische“ Variante, es wird ein Link mit dem Namen „Linkname“ erzeugt.
<code><ext name="extension_name"/></code>	Der von einer Extension erzeugte Code wird eingefügt
<code><![CDATA[(HTML-Inhalte)]]></code>	Alles was zwischen den eckigen Klammern steht wird später vom Processor nicht verarbeitet, sondern direkt an den Browser weitergereicht. Tags sollten aus Kompatibilität in dem Bereich unbedingt als <code>
</code> geschrieben werden!
<code></code>	Verlinkt die Grafik dynamisch, Grafiken werden im Sitefile definiert

Eine jede Section endet mit dem `</sect>` Tag.

Verwendung von XSS

Mit Hilfe von XSS Tags ist es möglich eigene Tags zu definieren, die dann in den Content Dateien verwendet werden können. Dazu ist es nötig sie am Anfang des Templates zu definieren:

```
<!-- INCLUDE STYLE SHEET style.xss -->
```

Ein XSS File könnte so aussehen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stylesheet>
  <template match="news"><table cellpadding="1" cellspacing="0" border="1"><apply-templates/></table></template>
  <template match="newsitem"><tr><apply-templates/></tr></template>
  <template match="date"><td><font face="Arial"><b><apply-templates/></b></font></td></template>
  <template match="newscontent"><td><font face="Courier"><u><apply-templates/></u></font></td></template>
</stylesheet>
```

Der header `<?xml version="1.0"?>` sollte der Parameter `encoding="ISO-8859-1"` vorhanden sein, da verschiedene andere XML Systeme diesen Parameter benötigen. `<apply-templates/>` bezeichnet die Stelle, an der später das zwischen dem Start- und dem Endtag stehende eingefügt wird. Es können dort auch weitere XSS Tags benutzt werden.

Mit folgendem Code und den oben definierten Style Sheets würde eine Newstabelle erstellt werden:

```
<news>
  <newsitem>
    <date>Montag, 29.4.02</date><newscontent>Dokumentation erstellt</newscontent>
  </newsitem>
  <newsitem>
    <date>Mittwoch, 1.5.02</date><newscontent>Heute ist Feiertag</newscontent>
  </newsitem>
</news>
```

Die selbst definierten Tags werden dann entsprechend vom CMS ersetzt.