Template Spezifikation

Einleitung:

Mein Template System besitzt viele Funktionen, über die man schnell den Überblick verlieren kann, deswegen sind die Funktionen hier zusammengefasst.

Namensdefinitionen:

In diesem Zusammenhang möchte ich ein paar Namen festlegen, die ich im späteren Verlauf verwenden werde.

Diese Liste ist gedacht zum immer wieder nachsehen, wenn ihr beim erstem durchlesen nicht alles versteht, dann ist das normal.

Template:

Ein Template ist eine Datei innerhalb des vorher definierten Template Ordners.

Diese Dateien sind vom Typ HTML und werden durch speziell definierte Ausdrücke erweitert werden, damit diese mit dynamischem Inhalt gefüllt werden können.

Node:

Eine Node ist ein Abschnitt innerhalb eines Templates, der bestimmte Teile des Templates gliedert.

Der Aufbau einer Node sieht im Allgemeinen so aus:

```
[!=Name der Node!]
    { Inhalt … }
[!/Name der Node!]
```

Wichtig ist, dass das schließende sowie das öffnende Tag denselben Namen tragen!

Tag:

Ein Tag ist eine vom Template-System definierte Variable, die später durch Inhalt ersetzt oder spezielle Funktionen des Template-Systems aufruft.

Der Aufbau eines Tags sieht im Allgemeinen so aus:

Allgemein:

```
Template:
#{Steuerzeichen}{Argumente}#
```

Beispiel:

```
Template:
#!test#
```

Die Bedeutung des Tags **#!test#** wird im späterem Verlauf noch geklärt.

Anmerkung:

Die Steuerzeichen der einzelnen Funktionen stehen später in Eckigen Klammern hinter dem Namen.

Die Funktionen des einzelnen Tags:

Die Tags haben viele Funktionen, Argumente und Eigenschaften, die ich in der folgenden Liste mal aufführen will.

Variablen Ausgabe [!]:

Argumente: 1 (String)

Die einfachste und doch am häufigsten genutzte Funktion des Template-Systems.

Mit dieser Funktion können Variablen, die über die PHP-Schnittstelle definiert wurden in das Template-System eingebunden werden.

Allgemein:

```
PHP:
$site->set("name", "wert");
Template:
#!name#
```

Die Node **#!name#** im oberem Beispiel, wird dann durch den vorher definierten Inhalt ersetzt.

In diesem Fall durch das Wort „wert“.

Sprachbedingte Ausgabe [%]:

Argumente: 1 (String)

Das CMS „HPClass“ ist im Grunde für Mehrsprachigkeit ausgelegt.

Dieses Konzept merkt man auch in den Funktionen des Template-Systems.

Die Funktion für die „sprachbedingte Ausgabe“ ersetzt bestimmte Variablen durch ihre entsprechenden Werte in der Übersetzungsdatei.

Die Übersetzungen müssen vorher in Dateien, mit entsprechendem Format, definiert worden sein, damit diese verwendet werden können.

Allgemein:

```
Template:
#%name#
```

Im obigem Beispiel wird die Variable **#%name#** durch das Wort oder den Satz ersetzte, das in der entsprechenden Sprachdatei den Schlüssel „name“ trägt.

Kommentare [/]:

Argumente: N/A

Wenn in einem Template Kommentare gesetzt werden sollen, die nicht an den Benutzer zurückgeliefert werden, wenn die Seite aufgerufen wird, dann kann dies mit dieser Funktion gelöst werden.

Allgemein:

```
Template:
#/  Hier folgen Definitionen  #
```

Der Text, der zwischen den beiden Rauten (#) steht, wird komplett entfernt und somit nicht an den Benutzer zurückgeliefert.

Variablen [V:]:

Argumente: 2 (String, ExtendedInput)

Im Template-System ist es möglich Variablen zu bestimmen, die später innerhalb desselben Templates oder im aufrufenden PHP-Script verwendet werden können.

Dieses Tag besitzt im Gegensatz zu den Tags oberhalb zwei Argumente.

Das erste Argument ist ein normaler String, der angibt, welchen Namen die Variable später tragen soll.

Das zweite Argument definiert den Inhalt.

Das spezielle an diesem Argument ist, dass dieses geparst wird.

Die Syntax für den Inhalt kann hier eingesehen werden.

Allgemein:

```
Template:
#V:{Name} : {Inhalt}#
```

Zu beachten ist, dass Name und Inhalt der Variablen durch " : " (Leerzeichen, Doppelpunkt, Leerzeichen) voneinander getrennt werden.

Beispiel:

```
Template:
#V:foo : "bar"#
```

Im obigen Beispiel wird die Variable mit dem Schlüssel „foo" auf den Wert „bar" gesetzt.

Vergleichs-Operation [=]:

Argumente: 4 (ExtendedInput, ExtendedInput, ExtendedInput, [ExtendedInput])

Mit dieser Funktion ist es möglich zwei Werte mit einander zu vergleichen.

Wenn diese beiden Werte übereinstimmen, dann wird der an dritter Stelle definierte Wert.

An vierter Stelle wird der Wert angegeben, der ausgegeben wird, wenn die beiden Werte nicht übereinstimmen.

Allgemein:

```
Template:
#={VergleichA} == {VergleichB} : {Ja} : {Nein}#
```

Beispiel:

```
Template:
#=seite == “4“ : “selected=selected“ : ““#
```

Im Beispiel oben wird überprüft ob die Variable, die entweder über PHP oder über ein Variablen Tag gesetzt wurde, mit dem Wert „4“ übereinstimmt, wenn ja, wird „selected=selected“ ausgegeben.

Bei Fehlern innerhalb der Argumente, werden statt {Ja} der Wert **[T]** und statt {Nein} der Wert **[F]** ausgegeben.

Achtung:

Keine typsichere Überprüfung

Bedingte Ausgabe [?]:

Benötigt optionales Kommandozeichen
Argumente: 2 bis 3 (ExtendedInput, ExtendedInput, [ExtendedInput])

Bei dieser Funktion handelt es sich um eine der am häufigsten genutzten Funktionen des Template-Systems.

Mit dieser Funktion ist es möglich einen Text auszugeben, wenn eine Bedingung zutrifft oder optional, wenn diese nicht zutrifft.

Im Gegensatz zu der oben beschriebenen Vergleichs-Operation können mit dieser Funktion deutlich mehr Anwendungsfälle abgedeckt werden.

Diese Funktion hat, wegen ihrer Vielseitigkeit ein weiteres Steuerzeichen, das nach dem Fragezeichen [?] des Tags kommt.

Dieses Steuerzeichen bestimmt, welcher Typ von Bedingung geprüft werden soll.

Allgemein:

```
Template:
#?{Steuerzeichen}{Bedingung} : {JA}#
```

```
Template:
#?{Steuerzeichen}{Bedingung} : {JA} : {Nein}#
```

Das Steuerzeichen kann verschiedene Ausprägungen haben, die in der folgenden Liste abgehandelt werden:

Boolsche Überprüfung [:]

Bei diesem Überprüfungstyp wird überprüft ob der Wert im Feld {Bedingung} **true** ist.

Es kann sich auch um den String **„true“** handeln.

Beispiel:

#?:show_mail : !Mail# *(Die Bedeutung von „!Mail“ wird im Bereich Erweiterte-Eingabe geklärt)*

Konfigurationswerte [=]

Bei diesem Überprüfungstyp wird überprüft ob der Konfigurationswert mit dem Schlüssel der im Feld {Bedingung} definiert ist, **true** ist.

Beispiel:

#?=allow_db : “” : “Keine Datenbank erlaubt“

Rechte Überprüfung []

Kommandozeichen weglassen

Bei diesem Überprüfungstyp wird überprüft ob der betrachtende Benutzer ein Recht mit dem Namen besitzt, das im Feld {Bedingung} definiert ist.

Es gibt für diesen Fall noch zwei spezielle Werte, die angegeben werden können und um die es sich nicht um Recht handelt.

Diese zwei lauten **login** und **superadmin**.

Diese zwei Bedingungen sind erfüllt, wenn entweder ein Benutzer eingeloggt ist oder wenn es sich bei dem betrachtendem Benutzer um einen Superadmin handelt.

Beispiel:

#superadmin : “ganz Geheim“#

#show_user : “” : %noright#

Funktionen [@]:

Argumente: 1 Funktion und beliebig viele Argumente (ExtendedInput)#

Bei dem Template System ist es möglich spezielle PHP Funktionen über ein Tag aufzurufen.

Den Rückgabewert dieser Funktionen wird dann dem Benutzer ausgegeben.

Allgemein:

```
Template:
#@{Funktionsname} : {Argument1} : … : {ArgumentN}#
```

Alternativ:

```
Template:
#@{Funktionsname}({Argument1}, …, {ArgumentN})#
```

Beispiel 1:

```
Template:
#@echo : “Hallo “ : username : “!“#
```

Beispiel 2:

```
Template:
#@echo(“Hallo “, username, “!“)#
```

Die Ausgabe würde lautet „Hallo {Wert der Variablen username}!“

Loops [L:]:

Argumente: 2 (Array, ExtendedInput)

Bei den Loop Befehl wird ein bestimmter Bereich oder Wert, für jeden Eintrag in dem übergebenem wiederholt.

Allgemein:

```
Template:
#L:{Array} : {Base}#
```

Für jeden Eintrag innerhalb des Arrays im Feld {Array} wird der Inhalt im Feld {Base} ausgegeben.

Innerhalb des Inhalts, der in {Base} angegeben wird, werden dieselben Ersetzungen angewandt, wie im gesamtem Template.

Deshalb muss es sich um das übergebene Array um ein zweidimensionales Array handeln mit folgendem Aufbau:

```
Array(
    Array(
        „Name“ => „Alice“,
        „Alter“ => „12“
    ),
    Array(
        „Name“ => „Bob“,
        „Alter“ => „15“
    ),
    …
}
```

Beispiel:

In diesem Beispiel ist eine Node definiert, mit folgendem Inhalt:

[!=LoopInhalt!]

#l:Name# ist #l:Alter# Jahre alt.

[!/LoopInhalt!]

Im Template steht folgendes wobei Array das oben definierte Array sei:

#L:array : !LoopInhalt#

Dann ist die Ausgabe:

Alice ist 12 Jahre alt.

Bob ist 15 Jahre alt.

Beachte:

Mit dem Tag #l:{Schlüssel des Arrays}# können die Werte des Arrays eingesetzt werden, wenn das Tag #!{Name}# benutzt wird, dann werden die Variablen genutzt, die vorher definiert wurden und nicht die Werte des Arrays.

Lightbox Popups [+]:

Argumente: 2 – 4 (String, ExtendedInput, String, String)

Mit dieser Funktion können LightBox-Popup-Links eingefügt werden.

Allgemein:

```
Template:
#+{Seite} : {Inhalt}#
```

```
Template:
#+{Seite} : {Inhalt} : {Argumente}#
```

```
Template:
#+{Seite} : {Inhalt} : {Argumente} : {Typ}#
```

Diese Tags sind meistens in den Templates bereits vordefiniert und sollten nicht ohne Wissen über deren Bedeutung geändert werden.

Diese Funktion wird selten verwendet.

Genauere Informationen sind diesbezüglich innerhalb dieser Datei nicht enthalten.

Erweiterte Eingabe:

In diesem Template System gibt es bestimmte Argumente, die immer gleich geparst werden, egal für welche Funktion diese verwendet werden, deswegen werden diese gesondert behandelt.

Einfacher String:

Ein String kann in folgender Form angegeben werden:

Allgemein:

```
Eingabe:
“{Text}“
```

Beispiel:

```
Eingabe:
“Hello World“
```

Sprachbedingte Ausgabe:

Ebenso wie als Tag existiert die sprachbedingte Ausgabe als Element der erweiterten Eingabe.

Allgemein:

```
Eingabe:
%{Name}
```

Funktionsaufrufe:

Als erweiterte Eingabe, kann auch eine Funktion hergenommen werden.

Der Rückgabewert dieser Funktion, kann dann z.B. als Argument für andere Funktionen hergenommen werden.

(Möglicherweise über Umweg über eine Variable)

Allgemein:

```
Eingabe:
@{Name}({Argumet1}, {Argument2}, … , {ArgumentN}
```

Beispiel:

```
Eingabe:
@echo(“Ihr Benutzername lautet: “, username, “!“)
```

Loop Variablen:

Nur gültig, wenn als {Base} Wert eines Loop Tags aufgerufen.

Allgemein:

```
Eingabe:
l:{Name des Schlüssels}
```

Zugriff auf Schlüssel des Arrays mit **#l:K#**

Siehe Loop.

Variablen:

Variablen, die im Template-System existieren (entweder über PHP oder über das Variablen Tag definiert), können eben in den erweiterten Eingaben genutzt werden.

Hierzu wird einfach der Name der zu benutzenden Variable angegeben.

Allgemein:

```
Eingabe:
{Name der Variable}
```

Nodes:

Nodes können ebenso übergeben werden.

Bei den Nodes gibt es zwei Modifikationen:

Methode 1:

```
Eingabe:
!{Name der Node}
```

Bei dieser Methode werden alle Tags innerhalb der Node geparst.

Methode 2:

```
Eingabe:
!!{Name der Node}
```

Bei dieser Methode werden keine Tags geparst.

Abschließende Worte und Beispiele:

Dieses Template-System wurde für das von mir entwickelte CMS „HpClass“ entworfen und speziell darauf angepasst.

Dieses System ist, ohne Änderungen an dem System auch nur für dieses verwendbar.

Im folgendem sind ein paar Beispiele, wie man das System verwenden kann.

Diese Beispiele sind ohne weiterführende Erklärung, da alle verwendeten Elemente oben im Detail beschrieben sind.

Beispiele:

```
#/ Der Pfad zu ihrer Homepage     #
#V:pageadress : @echo("http://", HTTP_HOST, PHP_SELF)#
#/ ------------------------------------------------------------------- #
#/ Der Text, den der Link zur Registrationsseite haben soll   #
#V:mailregistertext : "Aktivieren"#
```

…

Is a in array?

#?:inArray("a", array) : "true" : "false"#

Is x in array?

#?:inArray("x", array) : "true" : "false"#

#@echo : !Test#

```
[!=Loop!]
        #l:name# <br />
        #l:V.name#<br />
        #l:K# <br />
[!/Loop!]
…
#L:array2 : !Loop#


[!=Config!]
        <tr class="pluginLine">
                <td>#!name#</td>
                <td>#!description#</td>
                <td>#!input#</td>
        </tr>

[!/Config!]

[!=Input-Option!]
  <option value="#!ID#" #?:selected : "selected=selected"#>#!value#</option>
[!/Input-Option!]
```

Komplettes Beispiel eines Templates:

(news.html im template/sites Ordner)

```
#!!NAME=news
#!!AUTOR=MRH

[!=Main!]

  #!News#

  <p align="center"><a href="index.php?site=news&limit=20"><b>#%morenews#</b></a>

  #?newswrite : WriteNews#
  #?newsedit : StartEditNews# </p>

[!/Main!]

[!=News!]

<div class="newsbox">
<div class="newsbox_info"><b>#!titel# #!level# #%by# <a
href=index.php?site=user&show=#!ersteller#>#!ersteller#</a></b></div>
<div class="newsbox_info_date"><b>  geschrieben am <i>#!datum#</i></b></div>
<div class="newsbox_inhalt">
#!Content#

<br>

#!EditNews# #!DeletNews#

</div>
</div>


[!/News!]


[!=LbSite-Edit!]

  <form method="POST" action="index.php?site=news">

    <p align="left">Überschrift:<br>
    <input type="text" name="newstitel" size="80" value="#!titel#"></p>
    <input type="hidden" name="newsid" size="80" value="#!site#">
    <p align="left">Datum:<br>
    <input type="text" name="newsdate" size="20" value="#!datum#" #?:datDisabled :
"DISABLED" : ""#></p>
    <input type="hidden" name="newstyp" value="Info" />
      <p align="left">
     Level
     <select name="newslevel">
       <option #="0" == level : "selected=selected" : ""#>0</option>
       <option #="1" == level : "selected=selected" : ""#>1</option>
       <option #="2" == level : "selected=selected" : ""#>2</option>
       <option #="3" == level : "selected=selected" : ""#>3</option>
     </select>
     <br>Die Berechtigungen können in der Seite "Rechte" geändert werden. Level 0
bedeutet öffentlich.</p>
    <p align="left">
    <textarea rows="15" name="newstext" cols="74" id="t1">#!newstext#</textarea>
    <button type="submit" name="#!sitename#"> <img src="images/ok.gif"> </button>
<button type="reset"> <img src="images/abort.gif"> </button>
  </form>


  <a href="#"  id="PL_link" onclick="$('picturelist').style.display='';
$('PL_link').style.display='none'; return false;"> Picture List öffnen</a>
```

```
60:
61:   <div class="picturelist" id="picturelist" style="display:none">
62:   <div class="picturelist_left" OnMouseOver="picturelist_on = true;
    picturelist_goleft();" OnMouseOut="picturelist_on = false;"
    onclick="picturelist_left()"></div><div class="picturelist_holder"
    id="picturelist_holder"></div><div class="picturelist_right"
    OnMouseOver="picturelist_on = true; picturelist_goright();"
    OnMouseOut="picturelist_on = false;" onclick="picturelist_right()"></div>
63:   </div>
64:
65:
66:   <script>
67:   picturelist_setdata(new Array (#!picturelist#));
68:
69:   picturelist_print();
70:
71:   </script>
72:
73:
74: [!/LbSite-Edit!]
75:
76: [!=LbSite-Del!]
77:
78:   <p id="highlight">Möchten Sie die Newsmeldung wirklich löschen?</p>
79:   <table width="100%">
80:   <tr valign="bottom">
81:   <td>
82:   <form method="POST" action="index.php?site=news">
83:     <p align="center"><input type="hidden" name="newsiddel" size="3"
    value="#!ID#"><input type="submit" value="Löschen" name="newsdel"></form>
84:   </td>
85:   <td>
86:   </td>
87:   </tr>
88:   </table>
89:   <b>ID:</b> #!ID#<br>
90:   <b>Newstitel:</b> #!titel#<br>
91:   <b>Ersteller:</b> #!ersteller#<br>
92:   <b>
93:
94:
95: [!/LbSite-Del!]
```