

Emplacements

Ordre de recherche des modèles:

- ./config/templates/MonModule
- ./themes/MonTheme/templates/modules/MonModule
- ./modules/MonModule/pntemplates

Emplacements de stockage des adaptations spécifiques d'un module pour un thème:

```

/themes/YT/templates/modules/MonModule/
pour les modèles modifiés d'un module
/themes/YT/style/MonModule/style.css
pour modifier la feuille de style du module
/themes/YT/templates/modules/MonModule/plugins/
pour les plugin et les extras
/themes/YT/templates/modules/MonModule/images/
pour modifier les images utiliser par le module
→ La même structure se retrouve dans le répertoire
config/templates ou le répertoire
modules/MonModule/template. Les modifications sont
alors appliquées pour tous les thèmes.

```

Appels

```

<!--[fonction]-->
<!--[fonction param1="val" param2=$var]-->
<!--[$variable]-->
<!--[$variable|modifier]-->
<!--[$variable|modifier:"valeur"]-->
<!--[$variable|phpFunction:"valeur"]-->

```

Assignment

```

<!--[assign var="name" value="exemple" ]-->

```

Backticks

Math simple:

```

<!--[assign var="name" value='$var1+$var2']-->

```

Combinaison de chaînes:

```

<!--[assign var="nom"
value="$var1 something `var2`"]-->

```

Conditions

```

<!--[if $nom == 'Fred']-->
    Bienvenue Monsieur
<!--[elseif $nom == 'Wilma']-->
    Bienvenue Madame
<!--[else]-->
    Bienvenue, quoi que vous soyez
<!--[/if]-->

```

Qualifieurs:

- \$a == \$b – égaux
- \$a != \$b – non égaux
- \$a > \$b – supérieur à
- \$a < \$b – inférieur à
- \$a >= \$b – supérieur ou égal à
- \$a <= \$b – inférieur ou égal à
- \$a === 0 – test d'identité
- ! \$a – négation
- \$a mod \$b – modulo
- \$a is div by 4 – divisible par
- \$a is even – est un nombre pair
- \$a is even by \$b – pair par groupement de niveau (?)
- \$a is odd – est un nombre impair
- \$a is odd by \$b – impair par groupement de niveau(?)

Boucles

```

<!--[section name=id loop=$variable]-->
    <!--[$variable[id]]-->
<!--[sectionelse]-->
    $variable empty!
<!--[/section]-->
Or:
<!--[foreach item="currentItem" from=$items]-->
    <!--[$currentItem]-->
<!--[foreachelse ]-->
    $item empty!
<!--[/foreach]-->

```

Math

```

<!--[math equation="x*y" x=$haut y=$large]-->
ou
<!--[math equation="x*y" x=$haut y=$large
assign="var"]-->
<!--[ $var ]-->
ou avec des parenthèses
<!--[math equation="((x+y)/z)" x=2 y=10 z=2]-->

```

Modificateurs

Quelques modificateurs utilisés couramment:

- default** – assigne la valeur par défaut à une variable
- nl2paragraphs** – remplace le marquage „nouvelle ligne“ (n) en paragraphe
- regex_replace** – recherche et remplace une expression régulière
- replace** – rechercher remplacer de base
- string_format** – transforme en chaîne comme sprintf() en php
- strip_tags** – enlève toutes les balises HTML
- truncate** – tronque une variable d'une longueur définie
- cat** – concatène une valeur avec une variable donnée
- wordwrap** – emballe une chaîne dans une largeur de colonne

Toutes les fonctions php peuvent être utilisés implicitement comme des modificateurs. Utilisez @phpfunc avec des tableaux.

Quelques modificateurs plus spécifiques:

- activatelinks** – change les URL présentent dans les chaînes en liens
- phtml** – convertit une définition de langue en chaîne de langue
- pnmodcallhooks** – appelle les extensions
- userprofilelink** – crée un lien vers le profil utilisateur
- userprofilelink:"css"** – ajoute class="css"
- userprofilelink:':img/profile.gif"** – Utilise profile.gif au lieu du nom d'utilisateur, sans class
- yesno** – renvoie Oui si var = 1 et Non si var = 0
- onlineoffline** – renvoie En ligne si var = 1 et Hors ligne si var = 0
- activeinactive** – renvoie Actif si var = 1 et Inactif si var = 0

Modificateurs de sécurité:

- pnvarprepfordisplay** – supprime tout le HTML
- pnvarprehtmldisplay** – preserve quelques balises HTML

Les deux préviennent l'injection de code destructif

pndate_format

pndate_format:"dateSpecifiers"

formate la date et l'heure d'une chaîne strftime():

- %a – abbr. nom du jour
- %A – nom du jour complet
- %b – abbr. Nom du mois
- %B – nom du mois complet
- %d – jour du mois (entre 01 et 31)
- %D – sous la forme %m/%d/%y (mois/jour/année)
- %e – jour du mois (entre 1 et 31)
- %H – heure (entre 00 et 23)
- %I – heure (entre 01 et 12)
- %j – jour de l'année (entre 001 et 366)
- %k – heure (entre 0 et 23)
- %l – heure (entre 1 et 12)
- %m – mois (entre 01 et 12)
- %M – minute en nombre décimal
- %n – caractère de renvoi à la ligne
- %p – soit "am" ou "pm" selon le cas
- %r – heure en format 12h (am. et pm)
- %R – heure au format 24h
- %S – second en nombre décimal
- %t – insère une tabulation
- %T – au format %H:%M:%S
- %u – jour de la semaine en nombre décimal
- %V – ISO 8601:1988 numéro de la semaine
- %x – date préférée pour la localisation courante
- %X – heure préférée pour la localisation courante
- %y – année (entre 00 et 99)
- %Y – année incluant le siècle
- %Z – fuseau horaire ou nom ou abbr.
- % – littéralement le caractère "%"

pndate_format:"":dateString"

Utilisez les standards de date/heure définis dans les fichiers de langues. Exemple pour le français:

- datebrief – %d %b %Y
- datelong – %A %d %B %Y
- datestring – %A %d %B @ %H:%M:%S
- datetimebrief – %d %b %Y - %l:%M %p
- datetimelong – %A %d %B %Y - %l:%M %p
- dateinput – %Y-%m-%d
- datetimeinput – %Y-%m-%d %H:%M
- timebrief – %l:%M %p

Exemples de Modificateurs

```

<!--[ $title|default:'no title' ]--> – utilise la
valeur par défaut si la variable est vide
<!--[ $title|truncate:40:'&hellip;' ]--> – limite
la var. à 40 car en la complétant par ...
<!--[ "%2.f"|sprintf:$var ]--> – applique la
fonction PHP à la variable PHP (le param. PHP est
devant le )
<!--[ $accountlinks|sort ]--> – utilise la fonction
PHP sort sur tous les éléments du tableau

```

Fonctions

```

<!--[pnvarcleanfrominput name="param"
assign="var"]--> – récupère la valeur d'un paramètre
de l'URL
<!--[pnusergetvar name="user_icq" uid=1]--> –
récupère une donnée pour un utilisateur (pas d'uid =
utilisateur actuel)
<!--[pnusergetidfromname name=$nom
assign="uid"]--> – renvoie l'ID pour un nom
d'utilisateur donné
<!--[pnimg src="image.png"]--> – renvoie une balise
d'image avec comme fichier la valeur de src
modname – nom du module (défaut – module courant)
width,height – si aucun défini, utilise ceux de l'image
alt – si pas défini, renvoie une chaîne vide
altml – l si vrai la chaîne alt est assumée comme étant
une constante ML
title – si pas défini, renvoie une chaîne vide
titleml – si vrai, la chaîne title est assumé comme une
constante ML
optional – Si défini, le plugin ne renverra pas d'erreur
si l'image n'est pas trouvé
default – si défini, une image par défaut sera utilisée si
l'image attendue n'est pas trouvé (le chemin complet
est requis)
set – si le modname est 'core' alors le paramètre set
est défini pour pointer sur le répertoire /images/
nostoponerror – si défini et qu'une erreur survient, il n'y
a pas de trigger_error, mais le retour de faux qui remplit
à la place pnimg_error
assign – si défini, le résultat sera assigné à une
variable
Tous les autres paramètres sont passés à la fonction du
module
<!--[pnml name="_EXAMPLESTRING"]--> – lit une
constante de langue
Paramètres valables:
name – nom de la constante à renvoyer
html – traite la définition comme du HTML
noprocess – si défini, aucun traitement appliqué à la
valeur de la constante
assign – si défini, le résultat est assigné à la variable
Tous les autres paramètres sont passés à la fonction du
module
<!--[pnmodurl modname="Name" type="user"
func="display"]--> – crée une URL pour une fonction
spécifique du module
Paramètres valables:
modname – nom du module (requis)
type – généralement soit 'user' soit 'admin' (par défaut
'user')
func – fonction du module (par défaut 'main')
fragment – le fragment à cibler à l'intérieur de l'URL
ssl – fixe la constante à null, true, false
append – ajoute une chaîne à la fin de l'URL
assign – si défini, le résultat est assigné à la variable
Tous les autres paramètres sont passés à la fonction du
module

```

Fonctions modules

```
<!--[pnmodfunc modname="Name" type="user" P
func="main"]-->
```

execute fonction pour un module (puser.php, padmin.php ...)

Paramètres valables:

modname – nom du module (requis)

type – généralement soit 'user' soit 'admin' (par défaut 'user')

func – fonction du module (par défaut 'main')

assign – si défini, le résultat est assigné à la variable

Tous les autres paramètres sont passés à la fonction du module

```
<!--[pnmodapifunc modname="Name" type="user" P
func="main"]-->
```

execute une fonction pour un module API Paramètres valables:

modname – nom du module (requis)

type – généralement soit 'user' soit 'admin' (par défaut 'user')

func – fonction du module (par défaut 'main')

assign – si défini, le résultat est assigné à la variable

Tous les autres paramètres sont passés à la fonction du module

```
<!--[include file="fichier.htm" var1="$array"]-->
Et les tableaux peuvent aussi être passés. Utilisez un
appel normal pour les appeler:
```

```
<!--[foreach from=$links item=l]-->
. do stuff ...
<!--[/foreach]-->
```

```
<!--[include
file="/usr/local/include/templates/header.tpl"]-
->
```

chemin absolu

```
<!--[includeP
file='file:/usr/local/include/templates/headerP
.tpl']-->
```

chemin absolu (même chose)

```
<!--[includeP
file='file:C:/www/pub/templates/header.tpl']-->
pour windows le chemin absolu doit utiliser le préfixe
"file:")
```

```
<!--[include file='db:header.tpl']-->
```

inclut à partir d'un modèle dont la source est "db"

```
<!--[include file="$module.tpl"]-->
```

inclut une \$variable comme nom de modèle

```
<!--[include file="$module.tpl"]-->
```

ne marche pas avec des guillemets simples car ce

n'est pas une variable de substitution

```
<!--[include file="$path/$module.$view.tpl"]-->
inclut plusieurs $variable pour nommer le modèle et son
chemin
```

```
function smarty_modifieur_NOM($params, &$smarty)
pour les modificateurs
function smarty_block_NOM($params, &$smarty)
pour les blocs
```

NOM doit être le même dans le nom de fichier et le nom de plugin (casse sensible!)

Devrait toujours contenir des paramètres:

assign – pour assigner, doit retourner une valeur à la variable

Plugin "Maison"

```
<?php
function smarty_TYPE_NOM($params, &$smarty) {

    //Mon code ici

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

Plugins Module additionnels

pnmodavailable – le module X est-il installé et actif?

pnmodgetinfo – récupère le nom affiché du module

pnmodgetname – récupère le nom du module actuel

pnmodgetvar – récupère une variable du module

pnmodishooked – Le mod. A étend-t-il (hook) le mod. B?

pndebug

La fonction la plus essentielle pour les designers Zikula:
<!--[pndebug]--> – ouvre une fenêtre indiquant toutes les variables et valeurs utilisables pour une page

Insertions

```
<!--[insert name="getstatusmsg"]-->
```

obtient et insert le dernier message de statut posté lors de cette session

Paramètres valables :

style, class – msg placé dans une balise div avec les attributs indiqués

tag – spécifie une balise span ou div

assign – assigne à une variable

```
<!--[insert name=setpagevar var="title"
value="mytitle"]-->
```

attribue une variable à une page spécifique

variables possibles:

title – fixe le titre html (le nom du site et le slogan sont ajoutés)

description – fixe la description de la balise meta

keywords – fixe les mots clés dans la balise meta

stylesheet – inclut une feuille de style dans l'en-tête

javascript – inclut un javascript dans l'en-tête

body – ajoute des attributs à l'ouverture à la balise body

rawtext – ajoute du texte brut dans l'en-tête

footer – ajoute du texte brut avant la fermeture de la balise body

exemples:

```
<!--[insert name=setpagevar var="stylesheet"
value="/themes/MyTheme/style/extrastyle.css"]-->
```

insère la feuille extrastyle.css dans l'entête HTML.

Valide le HTML ou XHTML selon les paramètres de theme.php

```
<!--[insert name=setpagevar var="javascript"
value="/javascript/ajax/prototype.js"]-->
```

insère prototype.js dans l'en-tête. Zikula élimine les appels redondant d'un même script

```
<!--[insert name=setpagevar var="description"
value=$text|truncate:'250']-->
```

insère les 250 premiers car. D'une variable dans la description meta

```
<!--[insert name=setpagevar var="body"
value='onload="onLoad()"']-->
```

ajoute l'attribut onload dans la balise body

Fonctions des thèmes

```
<!--[title]-->
```

généralement le titre de la page constitué par le nom du site et le slogan.

Paramètres valables :

separator – les éléments du titre peuvent être séparés par une chaîne (optionnel: default ':')

noslogan – slogan n'est pas ajouté si true

nositenam – sitenam n'est pas ajouté si true

pager (numeric)

```
<!--[pager rowcount="400" limit="50"]-->
```

ajoute une pagination numérotée dans le modèle

Paramètres valables :

rowcount – nombre total d'items à paginer dans pour une page (s'il s'agit d'un tableau, une liste) ce total sera utilisé pour répartir la pagination.

limit – nombre d'items par page (si <0 illimité)

posvar – nom de la variable contenant la position (en général "offset")

display – soit 'page' ou 'startnum'. page = 1, 2, 3, 4, ...)

startnum = (1, 11, 21, 31, 41, ...)

anchorText – texte optionnel pour les ancres

hypertextes (ex: 'comments' pour l'ancre #comments)

(défaut: ")

maxpages – optionnel : nombre maximum de pages

affichées (les autres sont masquées/supprimées

(défaut: 0 = voir toutes les pages)

class – optionnel : classe s'appliquant au conteneur de la pagination (défaut : pn-pager)

processDetailLinks – should the single page links be processed? (default: false if using pagerimage.html, otherwise true)

template – optionnel, nom d'un fichier modèle

includeStylesheet – utilise une feuille de style prédéfinie (Défaut = yes).

Modèles déjà valables:

pagercss2.html

pagercss.html

pager.html

pagerimage.html

pagerintervals.html

pageritems.html

pagerjs.html

Bien sûr vous pouvez ajouter les vôtres

Pagination (alphabétique)

```
<!--[pagerabc posvar="letter" ]-->
```

affichage d'une pagination alphabétique

Paramètres possibles:

names – valeurs à utiliser (dans le tableau ou le csv) ex: "A;B;C;D" etc...

separator – chaîne séparant les lettres ex: "|" qui

donne |A|B|C|D|

posvar – nom de la variable indiquant la position

d'affichage des données, en général "letter"

forwardvars – virgule- point-virgule- ou espace

Fonctions utilisateur

Fonctions relatives à l'utilisateur (user)

pnusergetidfromname – récupère le pseudo avec UID

pnusergetlang – langue de l'utilisateur

pnusergettheme – thème de l'utilisateur

pnusergetvar – détails du profil avec UID

pnuserloggedin – le visiteur est-il loggué?

Fonctions Blocs

Les fonctions Blocs s'incluent dans les modèles de bloc et opèrent sur les contenus de ceux-ci.

```
<!--[securityutil_checkpermission_blockP
component="News::" instance="::" P
level="ACCESS_COMMENT"]-->
```

permet de faire des choses en donnant les permissions

```
<!--[/securityutil_checkpermission_block]-->
```

Les permissions ne sont pas définies au niveau du module. Vous pouvez en définir certaines pour les utiliser dans le modèle ou le module de permissions.

```
<!--[literal]-->
```

ce contenu ne sera pas interprété par le moteur de rendu des modules

```
<!--[/literal]-->
```

Ce marquage de bloc est souvent utilisé si vous avez besoin de commenter des définitions particulières de feuille de style pour la compatibilité avec Internet Explorer.

```
<!--[php]-->
```

echo("Je suis un nouveau codeur");

```
<!--[/php]-->
```

L'indication de code PHP est utilisé pour inclure du code php brut dans le modèle. Mais ce n'est pas une solution très propre. Idéalement on crée des plugins php pour faire ça!

```
<!--[include file="fichier.htm"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

```
<!--[include file="fichier.htm" var1="NEW"]-->
```

Inclut le fichier et lui passe une variable. Utilisez un appel normal pour afficher cette variable

```
<!--[$var1]-->
```

```
<!--[include file="fichier.htm" var1="$array"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

```
<!--[include file="fichier.htm" var1="NEW"]-->
```

Inclut le fichier et lui passe une variable. Utilisez un appel normal pour afficher cette variable

```
<!--[$var1]-->
```

```
<!--[include file="fichier.htm" var1="$array"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

```
<!--[include file="fichier.htm" var1="NEW"]-->
```

Inclut le fichier et lui passe une variable. Utilisez un appel normal pour afficher cette variable

```
<!--[$var1]-->
```

```
<!--[include file="fichier.htm" var1="$array"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

```
<!--[include file="fichier.htm" var1="NEW"]-->
```

Inclut le fichier et lui passe une variable. Utilisez un appel normal pour afficher cette variable

```
<!--[$var1]-->
```

```
<!--[include file="fichier.htm" var1="$array"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

```
<!--[include file="fichier.htm" var1="NEW"]-->
```

Inclut le fichier et lui passe une variable. Utilisez un appel normal pour afficher cette variable

```
<!--[$var1]-->
```

```
<!--[include file="fichier.htm" var1="$array"]-->
```

Inclut le modèle fichier.htm dans le modèle courant à partir du répertoire du modèle courant.

Plugins

Nommage des plugins:

fonction.NOM.php – pour les fonctions

modifieur.NOM.php – pour les modificateurs

block.NOM.php – pour les blocs

Nommage des fonctions:

function smarty_function_NOM(\$params, &\$smarty)

pour les fonctions

délimitant les variables des POST et GET à mettre dans les liens de pagination. Si non-défini, toutes les variables sont suivies.

additionalvars – virgule, point-virgule, espace délimitant une liste de variables additionnelles et leurs variable à utiliser dans les liens. Ex: "foo=2,bar=4"

class_num – classe pour les liens de pagination (<a> tags)

class_numon – classe de style pour la page active

printempty – print empty sel ('-')

CSS Classes

Classes communes

.pn-hide – équivalent de display:none

.pn-clearfix – utilisé pour éviter le diabolique

clear:both!

Classe pour les tableaux

.pn-datatable – uniformise la présentation des tableaux

<tr class="<!-- [cycle values="odd,even"] ->">

pour permettre un affichage alterné du fond ou style d'une ligne selon si elle est pair ou impair

<th class="pn-sortable"> pour permettre le tri par colonne à l'aide de javascript (prototype!)

Classes pour les formulaires

.pn-form – utilisé pour uniformiser la présentation des formulaires

.pn-formrow – utilisé pour styliser un div contenant une étiquette et un contrôle

Licence

```
/**
 * Zikula Application Framework
 *
 * @copyright (c) 2001, Zikula Development Team
 * @link http://www.zikula.org
 * @license GNU/GPL -?
http://www.gnu.org/copyleft/gpl.html
 */
```