Moteur LiteTemplate V.1.9

Telnes 2007-07-15
telness@hotmail.com

Table des matières

I. Introduction

LiteTemplate est une Classe qui permet d'interpréter des fichiers dit Template. Un fichier Template est un fichier html et seulement html à la seule différence qu'il y a des balises particulières aux endroits ou le texte va être dynamique.
LiteTemplate à été développé dans le souci d'être simple et avec un minimum de fonction. En effet l'idée n'est pas de réécrire un nouveau langage à l'intérieur même des fichiers Templates. De toute façon on s'aperçoit assez vite que peut de fonction nous suffise pour réaliser un tas de choses.
Pour utiliser liteTemplate il faut maîtriser un minimum de HTML et de PHP.

II. Apprendre par l'exemple

a) Débutons par un exemple simple

Exemple1.php

```
<?php
include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file('exemple1.tpl');

$newtpl -> assign(array(
              'TITRE'=>'Mon premier template',
              'HEAD'=>'message entete',
              'IMAGE'=>'ornithorynque_reduit.jpg',
              'COLOR'=>'#115588'
              ));
$newtpl->view();
?>
```

Include_once 'liteTemplate.Class.php' nous permet d'inclure le fichier litetemplate.Class.php qui contient notre classe. Ici il se situe dans le même répertoire que notre fichier exemple1.php.

$newtpl = new liteTemplate() nous permet de créer un nouvel objet liteTemplate qui s'appelle $newtpl.

$newtpl -> file('exemple1.tpl') nous permet de stipuler quel fichier tpl est associé à notre code. Ici on lui associe exemple.tpl.

$newtpl -> assign(array('TITRE'=>'Mon premier template',HEAD'=>'message en tete', 'IMAGE'=>'ornithorynque_reduit.jpg', 'COLOR'=>'#115588')) nous permet de définir les variables que nous allons remplacer dans le fichier tpl.

$newtpl->view() nous permet d'afficher le résultat.

Exemple1.tpl

```
<html>
<head>
      <title>{$TITRE}</title>
</head>

<body bgcolor="{$COLOR}">
  {$HEAD}
  <br>
  <img src="{$IMAGE}" align="center">

</body>
</html>
```

Les balises {$*TEXT*} correspondent aux variables définies dans la page exemple1.php. Lorsque que l'on exécute le fichier exemple1.php il va remplacer les balises par les valeurs que l'on à définit.

b) Exemple de balise HTML <select>

Une méthode permet de gérer des balises HTML select facilement.

Exemple2.php

```
<?php
include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file('exemple2_1.tpl');

$myselect = array('toto', 'titi', 'tata', 'tutu');

$newtpl -> htmlSelect('mon_select',$myselect);

$newtpl -> view();
?>
```

La méthode **htmlSelect(**'mon_select',$myselect**)** permet de passé à la balise template le tableau à mettre dans le select qui aura pour attribut id la valeur mon_select.
On peut ajouter un paramètre supplémentaire pour stipuler quel champ va être sélectionné.

htmlSelect("mon_select",$myselect, '1') va sélectionner titi**.** Un 4 ème attribut est disponible pour cette objet et permet d'ajouter des action javascript. Par exemple si l'on vauex ajouter une action onclick il suffit de faire :
htmlSelect('mon_select',$myselect,','onclick="mafonction()"')

Exemple2.tpl

```
<html>
<head>
      <title>Boucle</title>
</head>

<body>

{HTMLSELECT id=mon_select}

</body>
</html>
```

La balise **{HTMLSELECT id=mon_select}** permet de dire dans le template qu'il va y avoir un select. On peut ajouter l'attribut selected pour stipuler quel valeur va être sélectionné, exemple **{HTMLSELECT id=mon_select selected=1}** va mettre titi en selected.
Le selected dans la balise est prioritaire sur celle définit dans l'objet.

c) Exemple de tableau dynamique

Dès que l'on à des données et des tableaux on fait des tableaux dynamiques, c'est indispensable pour un affichage correct. Pour cela on va s'intéresser aux fichiers exemple2.php et exemple2.tpl

Exemple3.php

```
<?php
Include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file('exemple2.tpl');

$newtpl->assignTag ('BALISE', '1',array('NUM'=>array(
                                        '1',
                                        '2',
                                        '3',
                                        '4'
                                        ),
                                         'PRENOM'=>array(
```

```
                                                'JOJO',
                                                'JIJI',
                                                'JAJA',
                                                'JUJU'
                                                )
                                                ));
$newtpl->view();
?>
```

Seule la fonction **`assignTag()`** est nouvelle. Elle prend comme argument le nom de la balise, son numéro et le tableau associé (explication des fonctions par la suite).

Exemple3.tpl

```
<html>
<head>
      <title>Boucle</title>
</head>

<body>

<table border="1">
{BALISE id=1}
<tr><td>{$NUM}</td><td>{$PRENOM}</td></tr>
{/BALISE}
</table>

</body>
</html>
```

Voilà le tour est joué, ça vous affiche un tableau de 4 lignes.
Vous pouvez trouver que le code devient vite lourd lorsque vous avez un gros tableau ! Rien ne vous empêche de mettre vos tableaux dans des variables :

```
<?
Include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file('exemple2.tpl');

$num = array('1', '2', '3', '4');
$prenom = array('JOJO', 'JIJI', 'JAJA', 'JUJU');

$newtpl->assignTag ('BALISE', '1',array('NUM'=> $num,
                                        'PRENOM'=>$prenom,
                                          ));
$newtpl->view();

?>
```

d) Exemple avec une requête MySQL

Nous allons montrer ici comment intégré dans une page des données issues d'une base MySQL.

Exemple4.php

```
<?php

include_once 'liteTemplate.Class.php';


/* connexion à la base */

mysql_connect('localhost', 'root', '');
mysql_select_db('mabase');

/* récupération de champs de la table */

$sql = 'SELECT id,photo FROM gallerie' ;
$reponse = mysql_query($sql) or die($sql.' : '.mysql_error());

/* on declare 2 tableaux */
$photo = array();
$id = array();

while ($donnees = mysql_fetch_array($reponse) ){
      /* on ajoute les valeurs dans nos tableaux */
      $id[] = $donnees['id'];
      $photo[] = $donnees['photo'];

}
/* on créer un tableau au format liteTemplate */
$global = array('ID'=>$id,'PHOTO'=>$photo);

/* gestion du template */
```

```php
$newtpl = new liteTemplate();

$newtpl -> file("exemple4.tpl");

$newtpl -> assignTag('BALISE', '1',$global);

$newtpl -> view();
?>
```

Pour récupérer les valeurs id et photo de notre table gallerie deux tableaux vont être nécessaire : $photo et $id. Ensuite dans le while nous récupérons les valeurs des id et photo que nous plaçons dans les tableaux précédemment définis.
Comme nous récupérons plusieurs valeur il faut utiliser la méthode assignTag() pour créer une boucle côté template. Cette méthode prend en argument le nom de la balise boucle ici BALISE, un id ici 1 et un tableau de clé tableau ! c'est pourquoi $global est créer. Attention il faut que tous les tableaux passés en valeur des clés aient autant de valeur !
Une fois toutes les étapes réaliser il nous suffit juste de passer $global à assignTag().

Côté template nous allons avoir quelque chose de cette forme :

Exemple4.tpl

```
<html>
<head>
      <title>Boucle</title>
</head>

<body>

<table border="1">
{BALISE id=1}
<tr><td>{$ID}</td><td><img src="{$PHOTO}"></td></tr>
{/BALISE}
</table>

</body>
</html>
```

e) Exemple de include

Les includes sont également indispensables. En effet ça permet d'alléger sont code et de créer des pseudos frames.
Nous allons voir le principe dans l'exemple5_1.php, include5_1.php et exemple5_1.tpl.

Exemple5_1.php

```php
<?php
include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file("exemple5.tpl");

$newtpl -> assignInclude("1");

$newtpl -> view();
?>
```

Exemple5_1.tpl

```
<html>
<head>
      <title>Include</title>
</head>

<body>

{INCLUDE id=1 file=include5.php}

</body>
</html>
```

Dans le fichier exemple5.tpl on a ajouté la balise **{INCLUDE id=1 file=include3_1.php}** qui va nous permettre de récupérer par le biais de **assignInclude("1")** le fichier à inclure. Le paramètre de la fonction correspond à l'id de la balise include.
Cette méthode est obsolète depuis la V1.9 ou la balise template **{INCLUDE file=include3_1.php}** dans le fichier template est suffisant. Il n'y a plus besoin de faire appel à une fonction dans le code php.

Include5_1.php

```
<?php
echo  "fichier inclus";

?>
```

Pseudo frame

Il existe une autre méthode pour inclure un fichier. En effet on n'a pas toujours envie de stipuler le fichier à inclure dans le template, si l'on veut modifier la valeur de la page dynamiquement (permet de gérer des pseudos frames).

Nous allons voir le principe dans l'exemple5_3.php, include5_3.php et exemple5_3.tpl.

Exemple5_3.php

```
<?php

include_once 'liteTemplate.Class.php';

$newtpl = new liteTemplate();

$newtpl -> file("exemple5_3.tpl");

$newtpl ->assignInclude("1","include5_3.php");

$newtpl ->view();
?>
```

Exemple5_3.tpl

```
<html>
<head>
      <title>Include</title>
</head>

<body>

{INCLUDE id=1}

</body>
</html>
```

Dans le fichier exemple5_3.tpl on a ajouté la balise **{INCLUDE id=1}** qui va nous permettre de récupérer par le biais de **assignInclude("1","include3_2.php ")** le fichier à inclure. Le paramètre de la fonction correspond à l'id de la balise include et au nom du fichier.

Cette méthode est à utiliser pour les include 'dynamique ' (pseudo frame par exemple) ou la page est modifier dynamiquement contrairement à la méthode précédente.

f) Exemple de cache

Depuis la version 1.9 la gestion du cache est intégré à la classe.

L'intérêt de mettre les fichier en cache est de ne pas les reparser le fichier template. En effet lors du première appel le fichier parser (par le moteur litetemplate) est conservé dans un dossier (par défaut _cache) sans aucun code template (uniquement HTML). Lors des autres appel de cette page c'est le fichier en cache qui est retourné ce qui fait un gain de temps et diminue la charge du serveur.

Exemple de cache 4

```
<?php

include_once 'liteTemplate.Class.php';

// html select value
$myselect = array('toto', 'titi', 'tata', 'tutu');


/* template begin */

//activate cache
//activation du cache
$newtpl->cache_activate = true;

$newtpl = new liteTemplate();

$newtpl -> file("exemple6.tpl");

$newtpl -> htmlSelect('mon_select',$myselect);

$newtpl -> view();

?>
```

Les anciennes méthodes utilisées étaient **getCache** et **putCache**, qui sont déprécié depuis la version 1.9.

Ici nous avons juste ajouter une ligne $newtpl->**cache_activate** = true; pour le caching des fichiers.
Les autres propriétés liées au cache sont :
cache_folder pour changer le nom du répertoire de cache (par défaut _cache)

cache_life pour mettre un temps de validité du fichier en cache (par défaut 1 min)

cache compression

Disponible depuis la version 1.9

La compression est appliquée au fichier mis en cache, pour un gain de place. (Gain de 40% dans l'exemple 7 sur la taille du fichier)

Exemple 7

```
<?php

include_once 'liteTemplate.Class.php';

// html select value
$myselect = array('toto', 'titi', 'tata', 'tutu');


/* template begin */

//activation du cache
$newtpl->cache_activate = true;

//activation de la compression des fichiers mis en cache
$newtpl->cache_compression = true;

$newtpl = new liteTemplate();

$newtpl -> file("exemple6.tpl");

$newtpl -> htmlSelect('mon_select',$myselect);

$newtpl -> view();

?>
```

III. Définition de la classe LiteTemplate

a) Variables (attributs) globales

$tplName contient le nom de la page tpl

$time qui contient le temps d'exécution total de la classe. Elle est accessible seulement après l'appel de view().

$tpl contient toute la page template appelé dans file().

$cache_folder contient le chemin vers le répertoire de cache. Par défaut est _cache/.

$cache_life contient la durée de vie d'un fichier en cache. En seconde.

$cache_activate permet l'activation du cache

$cache_compression active la compression des fichiers en caches

$debug active le debug

$version à la version du moteur litetemplate

//Privat
$error[] est un tableau qui contient des information utile pour le débugge.

$cache_isExpired

b) Fonction (méthode) de la Classe

file()

none file*(file)*
Cette fonction permet de récupérer le fichier template qui va nous servir de squelette pour nos variables.

Exemple :

```
//on créer un objet template
 $newtpl = new liteTemplate() ;

//on appel notre fichier tpl
$newtpl  -> file("MonFichier.tpl" ) ;
```

assign()

none assign*(array)*
Cette fonction permet de passer un tableau array(key1=>valeur1,key2=>valeur2,…) ou les key correspondent aux balises dans le fichier template qui seront remplacées par les valeurs. Si une balise apparaît plusieurs fois, elles seront toutes remplacées.

Exemple :

```
//on créer un objet template
```

```
$newtpl = new liteTemplate() ;

//on créer un tableau avec nos clés valeurs
$array = array("NOM"=>"jojo","PRENOM"=>"titi") ;

$newtpl -> assign( $array ) ;
```

htmlSelect()

none htmlSelect(string,array,[string])
Permet de réaliser une balise html select (<select name= monSelect><option value=1>toto … </select>)
Le dernier paramètre est facultatif et permet de stipuler quel élément sera selectionné à l'affichage.

Exemple :

```
$myselect = array('toto', 'titi', 'tata', 'tutu');

$newtpl -> htmlSelect("mon_select",$myselect);
```

assignTag()

none assign_balise(*string* , *int* , *array*)
Permet de réaliser des boucles pour réaliser des affichages dynamiques avec les valeurs entre les balises. Le tableau passé en argument est du type clé=>array().

Exemple :

```
$newtpl->assign_balise("BALISE","1",array( "VAR"=>array(
                                            "1",
                                            "2",
                                            "3",
                                            "4"
                                            ),
                                             "NOM"=>array(
                                            "jojo",
                                            "toto",
                                            "nono",
                                            "bobo"
                                            ),
                                             "PRENOM"=>array(
                                            "gégé",
                                            "gogo",
                                            "gaga",
                                            "gigi"
                                            )));
```

ps : il faut qu'il y est le même nombre d'éléments dans tout les sous tableaux, sinon il y a un message d'erreur dans la variables error[] ;

assignInclude()

none assign_include(*int, string*)
Cette fonction permet d'inclure un fichier dans le template grâce à la balise
{INCLUDE id=1 file=fichier.php}
le numéros id est modifiable et c'est lui que l'on passe en argument de la fonction assign_include().

view()

string view()

Permet d'afficher le fichier final, après parseage des balises.

Exemple :

$newtpl -> **view()** ;

returntpl()

string returntpl()

Permet de retourner le contenu de la variable $tpl.

getError()

string getError()

Retourne les messages d'erreurs obtenus.

getCache() [deprecate] ::privat

string getCache(*string cachename*)

Retourne le contenu du cache ou false en cas d'échec.

putCache() [deprecate] :: privat

bol putCache(*string cachename* ,*string content*)

Met le contenu de content dans le cache. Retourne true en cas de succès et false sinon.

version()

string version()

Retourne les informations sur la version de liteTemplate

findTag()

[Fonction interne à la classe **:: Privat**]

array find_balise(*string* , *int*)
Cette fonction permet de récupérer une zone de texte entre 2 balises.
Une balise étant définie par un nom et un id {BALISE id=1} par exemple, et il ne faut pas oublier de la fermer, {/BALISE}

checkArray()

[fonction interne à la classe :: **Privat**]

bolean verif_tab(*array*)
Test si il y a le même nombre d'éléments dans les sous tableaux
array(key=>array(),key=>array()...)

getIncludeContents()

[fonction interne à la classe :: **Privat**]

String getIncludeContents(*file*)
Permet de récupérer la sortie d'un fichier

clearTag()

version 1.4
[fonction interne à la classe :: **Privat**]

string clearTag(none)
Supprime toutes les balises non utilisées, elle est appelée dans view(). Les balises supprimées sont de la forme {$MABALISE} et les balises dynamiques de la forme {MABALISE id=45} mon texte {/MABALISE}

IV. Contact

Pour me contacter envoyer un mail à telness@hotmail.com
Merci de me le signaler si vous faites des modifications, si elles sont bénéfiques ça serait dommage de ne pas les ajouter dans la Classe.
Si des personnes sont intéressées pour continuer le projet contactez-moi.