Virtual Template 1.3 - Documentation

par François Campana (Projet VTemplate)
Jean-Baptiste Thiebaut (Projet VTemplate)

Date de publication : 26/03/08

Dernière mise à jour :

VTemplate est un moteur de templates pour PHP.

I - Présentation du Projet
II - Les fichiers templates
II-A - Déclaration d'une variable
II-B - Déclaration d'une zone dans un fichier template
III - Fonctions de VTemplate
III-1 - Open() : Ouverture d'un fichier template
III-2 - SetVar() : Valorise une variable d'une session d'une zone
III-3 - SetVarF() : Remplace une variable d'une session d'une zone par le contenu d'un fichier
III-4 - SetGlobalVar() : Valorise une variable dans l'ensemble d'un fichier ou d'une zone (et sous zone)
III-5 - AddSession() : Ajouter une session
III-6 - CloseSession() : Fermer une Session
III-7 - NewSession() : Création automatique d'une session
III-8 - isCached() : Vérifie la validité du cache
III-9 - Parse() : Génère le code d'un fichier template et valorise la variable d'un autre fichier avec ce code généré
III-10 - Display() : Génère et Affiche les résultats

I - Présentation du Projet

Les templates consistent à séparer le code source de "l'apparence" du site. Il vous permet de créer plusieurs interfaces pour votre site. Vous pouvez par exemple utiliser les templates pour créer un site multi-langue.

Il existe de nombreux scripts (FastTemplate, ModeliXe, *etc*.) mais aucun ne nous a apporté réellement ce que nous recherchions.

Quand nous avons conçu la classe, nous voulions qu'elle soit très facile d'utilisation sans pour autant négliger les performances (nous n'avons pas réellement procédé à un benchmarking concernant VTemplate, mais nous avons testé quelques exemples avec FastTemplate. Les résultats se sont révélés plutôt encourageants pour VTemplate.

Le projet repose sur 2 principes :

- Les zones ;
- Les variables.

Les zones sont des "espaces" dans votre fichier template (ou même dans un fichier HTML) qui délimitent le code HTML que vous souhaitez répéter, changer...

Il n'y a pas de restriction concernant le nombre de zones et de variables que vous souhaitez utiliser.

cf. la section suivante pour plus de détails concernant l'utilisation de Virtual Template.

La classe utilise le principe de sessions. C'est-à-dire qu'une fois la structure parsée, vous pouvez créer une session ("instance") d'une zone dans laquelle vous accédez aux sous-zones (qui fonctionnent comme les zones dites parentes) et aux variable de la zone. Regardez les exemples fournis dans le package (repertoire **exemples**). Au moment de la fermeture des sessions, le code est généré. Vous ne pouvez donc plus modifier les valeurs des variables. Recréez une session avec de nouvelles valeurs ; cela génèrera du nouveau code *etc*.

Depuis la version 1.2, VTemplate gère plusieurs fichiers en même temps. Le fonctionnement de chaque fichier est le même : il est composé de zones et de variables que l'on peut manipuler, remplir... Une nouvelle fonction, permettant la liaison de données, a été implémentée : **Parse()** permet de récupérer le contenu d'une session d'un fichier (et par conséquent un fichier entier) et l'afficher (le stocker) dans une variable d'un autre fichier. Regardez la partie III de ce document pour savoir comment utiliser cette fonction.

II - Les fichiers templates

II-A - Déclaration d'une variable

Les variables sont déclarées généralement dans des **zones** (voir ci-dessous). Vous pouvez néanmoins utiliser des variables sans déclarer de zone. Dans ce cas, consultez la documentation d'**AddSession()** et de **NewSession()**.

Syntaxe :
```
{#NomDeLaVariable}
```

*la syntaxe **{NomDeLaVariable}** a été abandonnée car elle possait des problèmes avec des feuilles de styles. Vous pouvez néanmoins revenir à cette syntaxe en modifant le **Define** correspondant au début de la classe :*

```
define("VARTAG", "{");
```

Caractères valides pour le NomDeLaVariable : tous sauf '{' , '#' '}' et '|'.

Une zone peut contenir autant de variables que vous le désirez.

II-B - Déclaration d'une zone dans un fichier template

Elle est basée sur les balises HTML :

- Balise d'ouverture : **<!--VTP_NomDeLaZone-->**
- Balise de fermeture : **<!--/VTP_NomDeLaZone-->**

***VTP_** et **/VTP_** sont obligatoires !*

Caractères valides pour le nom de la zone : tous sauf '{' , '#' '}' et '|'. Vous pouvez déclarer autant de sous-zones que vous le désirez. Entre les balises de zones, vous pouvez placer du code HTML et des variables. Une zone peut ne pas contenir de variables (*cf.* **AddSession()** et **NewSession()** du chapitre suivant).

```
<!--VTP_zone1-->
    <table width=100>
    <!--VTP_zone2-->
        <tr>
        <td>{#i}</td>
        </tr>
    <!--/VTP_zone2-->
    </table>
<!--/VTP_zone1-->
```

III - Fonctions de VTemplate

VTemplate est une classe PHP. Elle nécessite cette déclaration :

```php
<?php

include("vtemplate.class.php"); // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object

// ... suite du programme
```

Liste des fonctions de VTemplate :

- Open()
- SetVar()
- SetVarF()
- SetGlobalVar()
- AddSession()
- CloseSession()
- NewSession()
- Parse()
- Display()

III-1 - Open() : Ouverture d'un fichier template

handle Open(*string* filename[, CACHED, *int* timelimit])

Ancien nom : int Open (string filename);

Description : Ouvre le fichier filename et parse ce fichier.

Retourne :

- Un handle en cas de succès. Ce handle identifie le fichier ouvert pour les fonctions **addSession()** et **SetVar()** (Remarque : On peut faire un rapprochement avec la fonction **fopen()** qui retourne un identificateur de fichier ouvert) ;
- -1 si une erreur c'est produite.

Paramètres :

- **filename** : chemin et nom du fichier à ouvrir et parser.

Paramètres (version 1.3 Cache Edition uniquement) :

- **CACHED** : constante indiquant si l'on veut mettre en cache le résultat du fichier ;
- **timelimit** : durée (en secondes) de validité du cache.

En cas de problème d'accès au fichier ou d'erreur de syntaxe dans ce fichier, le parse est arrêté et l'erreur rencontrée est affichée.

Utilisation simple :
```
$handle = $vtp->Open("nomfichier.vtp");
```

Utilisation avec cache :
```
$handle = $vtp->Open("chemin/nomfichier", CACHED, 3600);
if ( !$vtp->isCached($handle) ){
    //code à executer pour générer le fichier
}

$vtp->Display($handle); // Affiche le résultat
```

III-2 - SetVar() : Valorise une variable d'une session d'une zone

int setVar(*handle* id, *string* zone_var, *mixed* val)

Ancien nom : int AddVal(string zone_var, mixed val)

Description : Valorise la variable 'var', de la session de la zone 'zone', avec 'val'.

zone_var s'écrit de cette façon :
```
nom_de_la_zone.nom_de_la_variable
```

Retourne :

- 1 si tout se passe bien ;
- -1 en cas de problème. L'exécution du script continue mais un message d'erreur est affiché.


Vous devez ouvrir une session pour pouvoir valoriser une variable.


*Si vous avez créé un fichier template avec uniquement des variables, vous pouvez directement (sans préciser de zone) valoriser ces variables avec **setVar()** sans passer par les fonctions **addSession()**, **closeSession()** ou **newSession()**.*

```
$vtp->setvar($handle, "nom_de_la_variable", $valeur);
```

Exemple :
```
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
$vtp->addSession($handle, "mazone");
// La zone 'mazone' contient une variable var
$vtp->setVar($handle, "mazone.var", $i);
$vtp->closeSession($handle, "mazone");

$vtp->Display();
```

III-3 - SetVarF() : Remplace une variable d'une session d'une zone par le contenu d'un fichier

int setVarF(*handle* id, *string* zone_var, *string* file_name)

Ancien nom : int AddValF(string zone_var, string file_name [, int handle])

Description : Valorise la variable 'var' de la session de la zone 'zone', avec 'val' par le contenu du fichier file_name.

zone_var s'écrit de cette façon :
```
nom_de_la_zone.nom_de_la_variable
```

Retourne :

- 1 si tout se passe bien ;
- -1 en cas de problème. L'exécution du script continue mais un message d'erreur est affiché.

Vous devez ouvrir une session pour pouvoir valoriser une variable.

*Si vous avez créé un fichier template avec uniquement des variables, vous pouvez directement (sans préciser de zone) valoriser ces variables avec **setVar()** sans passer par les fonctions **addSession()**, **closeSession()** ou **newSession()**.*

```php
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
$vtp->addSession($handle, "mazone");
// La zone 'mazone' contient une variable header
$vtp->setVarF($handle, "mazone.header", "header.htm");
$vtp->closeSession($handle, "mazone");

$vtp->Display();
```

III-4 - SetGlobalVar() : Valorise une variable dans l'ensemble d'un fichier ou d'une zone (et sous zone)

int setGlobalVar(*handle* arg, *string* zone_var, *mixed* val)

Description : Valorise la variable 'var', de la zone 'zone' et ses sous zones, avec 'val'.

zone_var s'écrit de cette façon :
```
nom_de_la_zone.nom_de_la_variable
```

Retourne :

- 1 si tout se passe bien ;
- -1 en cas de problème. L'exécution du script continue mais un message d'erreur est affiché.

Arg désigne le **handle** du fichier sur lequel vous souhaitez utiliser cette fonction. Vous pouvez également valoriser l'ensemble des fichiers VTP en utilisant la constante **ALL**.

```php
p->setVar($handle, "var", $i);
// remplace toutes les variables nommées "var" dans le fichier indiqué par le handle. Ce fichier
 doit avoir été préalablement ouvert.

$vtp->setVar(ALL, "var", $i);
// remplace toutes les variables nommées "var" dans tous les fichiers. Il n'est pas nécessaire que
 les fichiers soient préalablement ouverts.

$vtp->setVar($handle, "zone.var", $i);
```

```
// remplace toutes les variables nommées "var" dans la zone "zone" et ses sous zone. Le fichier
 indiqué par le handle doit avoir été préalablement ouvert.
```

III-5 - AddSession() : Ajouter une session

int addSession(*handle* id, *string* nomzone[, CACHED, *int* timelimit, *int* numsession])

Description : Crée une session sur la zone nomzone.

Retourne :

- 1 si tout se passe bien ;
- Retourne 0 si le cache est toujours valable ;
- -1 en cas de problème. L'exécution du script continue mais un message d'erreur est affiché.

Paramètres (version 1.3 Cache Edition uniquement) :

- **CACHED** : indique si l'on veut mettre dans le cache la session demandée ;
- **timelimit** (int) : durée (en secondes) de la validité du cache ;
- **numsession** : cette variable permet de mettre en cache une certaine session d'un zone ; par exemple pour une zone "test" générée 5 fois (=> 5 sessions), on peut mettre en cache les sessions 2 et 5 ;

Pour afficher une zone sans variable, il suffit d'ouvrir et de fermer une session.

*Si vous avez créé un fichier template avec uniquement des variables, vous pouvez directement (sans préciser de zone) valoriser ces variables avec **setVar()** sans passer par les fonctions **addSession()**, **closeSession()** ou **newSession()**.*

Version normale :

```
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
for($i=1; $i<5; $i++){
    $vtp->addSession($handle, "mazone");
    // La zone 'mazone' contient une variable var
    $vtp->setVar($handle, "mazone.var", $i);
    $vtp->closeSession($handle, "mazone");
}

$vtp->Display();
```

Version avec cache :

```
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
for($i=1; $i<5; $i++){
    $test = 1;
    if ($i==2) {
        $test = $vtp->addSession( $handle, "mazone", CACHED, 3600);
    }
    else {
        $vtp->addSession($handle, "mazone");
    }

    if ($test){
```

Version avec cache :
```
            // La zone 'mazone' contient une variable var
            $vtp->setVar($handle, "mazone.var", $i);
        }
        $vtp->closeSession($handle, "mazone");

    }

    $vtp->Display();
```

III-6 - CloseSession() : Fermer une Session

int closeSession(*handle* id,*string* nomdelazone)

Description : Ferme la session nomzone et génère le code.

Retourne :

- Retourne 1 si tout se passe bien.
- Retourne -1 en cas de problème.L'execution du script continue mais un message d'erreur est affiché.

Lors de la fermeture d'une zone, ses sous zones sont automatiquement réinitialisées à NULL.

*Si vous avez créé un fichier template avec uniquement des variables, vous pouvez directement (sans préciser de zone) valoriser ces variables avec **AddVal()** et **AddValF()** sans passer par les fonctions **addSession()**, **closeSession()** ou **newSession()**.*

Exemple 1 :
```
    <?php
    include("vtemplate.class.php");  // Inclusion du fichier
    $vtp = new VTemplate; // Déclaration de l'object
    $handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
    for($i=1; $i<5; $i++){
        $vtp->addSession($handle, "mazone");
        // La zone 'mazone' contient une variable var
        $vtp->setVar($handle, "mazone.var", $i);
        $vtp->closeSession($handle, "mazone");
    }

    $vtp->Display();
```

Exemple 2 :
```
    <?php
    include("vtemplate.class.php");  // Inclusion du fichier
    $vtp = new VTemplate; // Déclaration de l'object
    $handle = $vtp->Open("test2.vtp"); // Dans le fichier test2.vtp, il y a une zone "mazone" ne
     contenant que du HTML
    $vtp->addSession($handle, "mazone");
    $vtp->closeSession($handle, "mazone");

    $vtp->Display();
```

III-7 - NewSession() : Création automatique d'une session

int newSession(*handle* id, *string* nomzone)

Description : Ferme automatiquement la session précédent de 'nomzone' (si elle existe) ainsi que celle de ses sous-zones, puis ouvre une nouvelle session sur 'nomzone'.

Retourne :

- 1 si tout se passe bien ;
- 0 si le cache est toujours valable.
- -1 en cas de problème. L'exécution du script continue mais un message d'erreur est affiché.
- Version avec Cache:

Paramètres (version 1.3 Cache Edition uniquement) :

- **CACHED** : indique si l'on veut mettre dans le cache la session demandée.
- **timelimit** (int): durée (en secondes) de la validité du cache.
- **numsession** : cette variable permet de mettre en cache une certaine session d'un zone ; par exemple pour une zone "test" générée 5 fois (=> 5 sessions) : on peut mettre en cache les sessions 2 et 5.

*Cette fonction réduit (un peu) les performances de la classe. Pour une exécution optimale, utilisez **addSession()** et **closeSession()**.*

Remarque: Pour afficher une zone sans variable, il suffit d'utiliser la fonction **NewSession()**.

Lors de la fermeture d'une zone, ses sous zones sont automatiquement réinitialisées à NULL.

*Si vous avez créé un fichier template avec uniquement des variables, vous pouvez directement valoriser ces variables avec **setVar()** sans passer par les fonctions **addSession()**, **closeSession()** ou **newSession()**.*

Exemple 1 :

```php
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp"); // Dans le fichier test.vtp, il y a une zone "mazone"
for($i=1; $i<5; $i++){
    $vtp->newSession($handle, "mazone");
    // La zone 'mazone' contient une variable var
    $vtp->setVar($handle, "mazone.var", $i);
}

$vtp->Display();
```

Exemple 2 :

```php
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test2.vtp"); // Dans le fichier test2.vtp, il y a une zone "mazone" ne
 contenant que du HTML
$vtp->newSession($handle, "mazone");

$vtp->Display();
```

III-8 - isCached() : Vérifie la validité du cache

int isCached(*handle* id)

VTemplate 1.3 Cache Edition uniquement !

Description : Vérifie si le fichier associé au handle **id** est caché et si son cache est encore valable. Le handle est celui retourné par la fonction **Open()**.

Retourne :

- 1 si le cache est encore valable ;
- 0 si le fichier n'est pas caché ou si le cache du fichier a expiré.

```php
<?php
include("vtemplate.class.php");  // Inclusion du fichier
$vtp = new VTemplate; // Déclaration de l'object
$handle = $vtp->Open("test.vtp", CACHED, 800); // Ouverture du fichier en mode cache.Le cache dure
 800 secondes
if( !$vtp->isCached($handle) ){
    $vtp->addSession($handle, "mazone");
    // La zone 'mazone' contient une variable header
    $vtp->setVarF($handle, "mazone.header", "header.htm");
    $vtp->closeSession($handle, "mazone");
}

$vtp->Display($handle);
```

III-9 - Parse() : Génère le code d'un fichier template et valorise la variable d'un autre fichier avec ce code généré

Parse (*handle* handle_destination, *string* zone_var, *handle* handle_source[, *string* zone])

Ancien nom : Parse_Ext()

Description : Cette fonction permet de gérer le multi-fichier.

Elle génère le code issu du parsing du template sur le fichier indiqué par le **handle_source** (et optionnellement sur une zone précise de ce fichier). Elle valorise ensuite la variable du fichier indiqué par le **handle_destination**, par le résultat de la génération à l'étape précédante.

Lors de la fermeture d'une zone, ses sous zones sont automatiquement réinitialisées à NULL.

copyright.vtp

```
==== debut fichier copyright.vtp =====<br>
<!--VTP_mazone-->
    debut zone "mazone" de copyright<br>
    {#var}<br>
    fin zone "mazone" de copyright<br>
<!--/VTP_mazone-->
==== fin fichier copyright.vtp =====<br>
```

main.vtp

```
==== debut fichier main.vtp =====<br>
<html>
bla bla bla bla

<!--VTP_mazone-->
    <br>dans une zone :<br>
```

main.vtp

```
        {#var2}<br>
    <!--/VTP_mazone-->
    ou directement dans le fichier main<br>
    {#copy}
    </html>
    ==== fin fichier main.vtp =====<br>
```

Version normale :

```php
    <?php
    include("vtemplate.class.php");  // Inclusion du fichier
    $vtp = new VTemplate; // Déclaration de l'object
    $main = $vtp->Open("main.vtp");
    $fichier2 = $vtp->Open("copyright.vtp");

    $vtp->addSession($fichier2, "mazone");
    // La zone 'mazone' contient une variable var
    $vtp->setVar($fichier2, "mazone.var", "999999999999");
    $vtp->closeSession($fichier2, "mazone");

    $vtp->addSession($fichier2, "mazone");


    $vtp->addSession($main, "mazone");
    $vtp->Parse($main , "mazone.var2", $fichier2, "mazone" );
    // le contenu de la zone "mazone" du fichier2 est copié dans la variable "var2" de la zone "mazone"
     du main
    $vtp->closeSession($main, "mazone");

    $vtp->Parse($main, "copy", $fichier2);
    // le contenu du fichier2 est copié dans la variable "copy" du main

    $vtp->Display(); // Affiche tout ce qui a été généré au-dessus
```

Version avec cache :

```php
    <?php
    include("vtemplate.class.php");  // Inclusion du fichier
    $vtp = new VTemplate; // Déclaration de l'object
    $main = $vtp->Open("main.vtp");

    $fichier2 = $vtp->Open("copyright.vtp", CACHED, 3600);

    if ( !$vtp->isCached($fichier2) ){
        $vtp->addSession($fichier2, "mazone");
        // La zone 'mazone' contient une variable var
        $vtp->setVar($fichier2, "mazone.var", "999999999999");
        $vtp->closeSession($fichier2, "mazone");
    }

    $vtp->addSession($main,"mazone");
    $vtp->Parse($main, "mazone.var2", $fichier2, "mazone" );
    // le contenu de la zone "mazone" du fichier2 est copié dans la variable "var2" de la zone "mazone"
     du main
    $vtp->closeSession($main, "mazone");

    $vtp->Parse($main, "copy", $fichier2);
    // le contenu du fichier2 est copié dans la variable "copy" du main

    $vtp->Display(); // Affiche tout ce qui a été généré au-dessus.
```

III-10 - Display() : Génère et Affiche les résultats

[*string*] Display([*handle* id][, *int* disp[, *string* zone]])

Ancien nom : [string] Afficher([int handle], [string zone]);

*Changement de l'ordre des paramètres par rapport à la fonction **Afficher()**.*

Description : Génère le code issu du parsing du template.

Si vous décidez que **disp=0**, alors **Display()** retourne le code généré mais ne l'affiche pas. Par défaut, **disp** vaut 1 et affiche le résultat.

La version 1.1.2 ajoute un nouveau paramètre. Il permet d'afficher une zone particulière. Cette fonctionnalité est utilisée pour gérer les pieds de page et les entêtes.

La version 1.3 ajoute un nouveau paramètre. Il permet de préciser quel fichier on doit afficher. Par défaut, la fonction **Display()** génère le code du premier fichier ouvert. Si vous souhaitez générer le code d'un autre fichier, vous devez préciser le **handle** de ce fichier.

```
$vtp->Display(); // Affiche tout ce qui a été généré au dessus.
$code = $vtp->Display(0); // Récupère le code généré sans l'affiché
$vtp->Display($handle3, 1, "FOO3"); // Affiche la zone FOO du fichier d'handle $handle3.
```