

# Gestion d'un système de template en PHP

par Adrien Pellegrini ([Page d'accueil](#))

Date de publication : 24 août 2006

Dernière mise à jour :

Création et explication d'un système de templates en PHP avec gestion de constantes avec ou sans boucle.

- 1 - Introduction
  - 1.1 - Remerciements
  - 1.2 - Problématique
  - 1.3 - Qu'est ce qu'un système de template ?
- 2 - Exemple : test.php et test.tpl
  - 2.1 - Simples variables
  - 2.2 - Variables dans une boucle
- 3 - La classe Template
  - 3.1 - Sa structure
  - 3.2 - Ses méthodes
    - Méthode \_\_construct()
    - Méthode simpleVar()
    - Méthode loopVar()
    - Méthode constantReplace()
    - Méthode parse()
- 4 - Méthode(s) supplémentaire(s)
  - 4.1 - Méthode includeFile()
- 5 - Conclusion
  - 5.1 - Épilogue
  - 5.2 - Liens
- 6 - Annexe
  - 6.1 - Structure de la classe (code PHP)
  - 6.2 - Structure avancée de la classe
  - 6.3 - Structure du tableau \$InfoTpl

## 1 - Introduction

### 1.1 - Remerciements

Tous mes remerciements à **Yogui** pour sa relecture.

### 1.2 - Problématique

 **Ce tutoriel nécessite PHP 5**

Ce tutorial a pour but de vous donner une piste sur la création d'un système de template.

Je vous conseille de le lire une fois tout en entier.

#### **Ceci pour plusieurs raisons**

- 1 Il est question de créer une classe qui permettra la gestion de template. Cette classe contiendra plusieurs méthodes qui auront chacune leur rôle. Afin de mieux comprendre pourquoi telle ou telle méthode existe, lisez le tout une première fois;
- 2 Certaines méthodes dépendent des autres.

Toutefois, certains points de ce tutorial peuvent ne pas être lus au premier passage. Par exemple, le point 3.4 Méthode loopVar() n'est pas utile au premier passage car pour bien le comprendre, il faut avoir compris les points 3.3 Méthode simpleVar(), 3.5 Méthode constantReplace() (la première partie de la fonction ligne 1 à ligne 15) et 3.6 Méthode parse().

### 1.3 - Qu'est ce qu'un système de template ?

Un système de template est un système permettant de séparer le code source d'une page (PHP) de son design (HTML).

#### **Ce système présente quelques avantages**

- 1 Pouvoir travailler sur le design sans modifier une ligne de code;
- 2 Pouvoir modifier le code sans se soucier du design;
- 3 Créer facilement plusieurs design pour le site;
- 4 Faciliter la maintenance du code;
- 5 Permet de gérer un cache.

## 2 - Exemple : test.php et test.tpl

Voici les fichiers d'exemples qui nous serviront de base pour les explications.

### 2.1 - Simples variables

Exemple 2.1 : test.php - source avec des variables simples (placées hors boucle)

```
<?php
include 'template.php';

// Instanciation de la classe
$t = new Template('test.tpl');

// Simple variable
$t->simpleVar(array(
    'WELCOME_MSG' => 'Bonjour !!',
    'GOODBYE' => 'Au revoir !!',
));
$t->parse();
?>
```

Exemple 2.2 : test.tpl - template avec des variables simples (placées hors boucle)

```
<body>
    {WELCOME_MSG} <br />
    {GOODBYE}
</body>
```

### 2.2 - Variables dans une boucle

Exemple 2.3 : test.php - source avec des variables placées dans une boucle

```
<?php
include 'template.php';

// Instanciation de la classe
$t = new Template('test.tpl');

// Variable avec boucle
$country_array = array('BE' => 'Belgique',
    'FR' => 'France',
    'ITA' => 'Italie',
);

foreach ($country_array as $id => $country) {
    $t->loopVar('country', array(
        'ID' => $id,
        'COUNTRY' => $country,
    ));
}
$t->parse();
?>
```

Exemple 2.4 : test.tpl - template avec des variables placées dans une boucle

```
<body>
```

**Exemple 2.4 : test.tpl - template avec des variables placées dans une boucle**

```
<!-- BEGIN country -->
boucle :
<b>{country.ID}</b> => {country.COUNTRY} <br/>
<!-- END country -->
</body>
```

## 3 - La classe Template

Je vais vous expliquer pas à pas la réalisation d'une classe qui gère un système de template.

### 3.1 - Sa structure

Ne vous attardez pas trop sur cette structure, c'est juste pour vous donner un aperçu de la classe avec ses différentes méthodes.



*Schéma - structure de la classe*

## 3.2 - Ses méthodes

### Méthode \_\_construct()

#### Cette méthode a pour but de

- Tester si le fichier .tpl joint au code PHP existe;
- Stocker le code HTML de la page test.tpl dans la variable \$this->page.

#### méthode \_\_construct()

```
function __construct($file) {
    // Teste si le fichier existe et si il est autorisé en lecture
    if(empty($file) or !file_exists($file) or !is_readable($file)) {
        // Si le fichier est inexistant pas : erreur
        die('Template error : file '.$file.' not found.');
```

### Méthode simpleVar()

#### Cette méthode a pour but de

- Récupérer les constantes/données du fichier test.php (exemple 2.1) des simples variables;
- Stocker les constantes/données dans le tableau \$this->infoTpl selon le schéma défini précédemment pour ce tableau.

#### méthode simpleVar()

```
function simpleVar($varArray = array()) {
    // Si le tableau est vide, on stoppe le script
    if (empty($varArray)) exit;

    // Parcours du tableau
    foreach ($varArray as $var => $data) {
        // Enregistrement dans le tableau $this->infoTpl
        $this->infoTpl['.'][][$var] = $data;
    }
}
```

### Méthode loopVar()

#### Cette méthode a pour but de

- Récupérer les constantes/données du fichier test.php (exemple 2.3) avec des variables placées dans une boucle;
- Stocker les constantes/données dans le tableau \$this->infoTpl selon le schéma précédemment défini pour ce tableau.

## méthode loopVar()

```
function loopVar($type, $varArray = array()) {
    // Si le tableau est vide, on stoppe le script
    if (empty($varArray)) exit;

    // Calcule le nombre de lignes dans le type courant
    // Si 0 ligne => 0
    // Si X lignes => X
    // Pourquoi X et non (X + 1) ?
    // -> Car on compte à partir de 0 donc X retourne toujours
    // le dernier id + 1
    $lastID = (count($this->infoTpl[$type]) != 0) ? (count($this->infoTpl[$type])) : 0;

    foreach ($varArray as $constant => $data) {
        $this->infoTpl[$type][$lastID][$constant] = $data;
    }
}
```

## Méthode constantReplace()

## Cette méthode a pour but de

- Remplacer '{CONSTANTE}' du fichier .tpl par sa donnée contenue dans le fichier test.php.

## méthode constantReplace()

```
function constantReplace() {
    // Parcours de tout le tableau $this->infoTpl
    foreach($this->infoTpl as $type => $info) {
        // Si le type est '.' càd
        // provient de la fonction simpleVar()
        // ou encore de constantes places hors-boucle
        if ($type == '.') {
            for ($i = 0, $imax = count($info); $i < $imax; $i++) {
                foreach ($info[$i] as $constant => $data) {
                    // Remplace {CONSTANTE} par les données correspondantes
                    // et mets jour le code HTML du fichier test.tpl
                    // stock dans $this->page
                    $data = (file_exists($data)) ? $this->includeFile($data) : $data;
                    $this->page = preg_replace('\{' . $constant . '\}', $data, $this->page);
                }
            }
        }
        // Sinon si le type est autre càd
        // provient de la fonction loopVar()
        // ou encore de constantes places dans une boucle
    } else {
        // Calcule la taille du tableau $info
        $infoSize = count($info);

        // Variable qui contiendra le code à la place de
        // <!-- BEGIN country -->
        // {country.ID} => {country.COUNTRY}
        // <!-- END country -->
        $block = '';

        // Parcourt le tableau $info
        for ($i = 0; $i < $infoSize; $i++) {
            // Encode les caractères spéciaux
            $page = htmlentities($this->page);

            // $page est une variable string
            // Remplit le tableau $infoArray ligne par ligne
            $infoArray = explode("\n", $page);
```

## méthode constantReplace()

```
// Suppression des espaces blancs avant/après
// dus aux indentations du code
for ($k = 0, $kmax = count($infoArray); $k < $kmax; $k++) {
    $infoArray[$k] = trim($infoArray[$k]);
}

// Récupération et formatage des tags
$startTag = '<!-- BEGIN '.$type.' -->';
$startTag = htmlentities($startTag);

$endTag = '<!-- END '.$type.' -->';
$endTag = htmlentities($endTag);

// Récupération de la clé des tags dans le tableau $infoArray
$startTag = (array_search($startTag, $infoArray)) + 1;
$endTag = (array_search($endTag, $infoArray)) - 1;
// Nombre de lignes entre les tags
$lengthTag = ($endTag - $startTag) + 1;

// Récupération de la portion du tableau
// délimite par les tags (tags non compris)
$blockTag = array_slice($infoArray, $startTag, $lengthTag);

// Remise en type 'string' et non plus 'array'
// Facilitera le remplacement des constantes par leurs données
$blockTag = implode("\n", $blockTag);

// Remplacement des constantes par leur données
foreach($info[$i] as $constant => $data) {
    $data = (file_exists($data)) ? $this->includeFile($data) : $data;;
    $blockTag = preg_replace("`{'.$type.'.'$constant.'}`", $data, $blockTag);
}

// Ajout des données à la variable block globale pour la boucle
// Ajoute \n ou pas et ajoute les données
// de la nouvelle boucle à la suite de $block
$block = ($block == '') ? $blockTag : $block."\n".$blockTag;
}

// Mise en tableau de $block
// Facilitera l'opération de reconstitution des tableaux
$block = explode ("\n", $block);

// Coupe du tableau en 2
// $firstPart = début du tableau -----> <!-- BEGIN country -->
// $secondPart = <!-- BEGIN country --> -----> fin du tableau
$firstPart = array_slice($infoArray, 0, $startTag - 1);
$secondPart = array_slice($infoArray, $startTag + $lengthTag + 1);

// Reconstitution du code source
// en insrant au milieu les données
$page = array_merge($firstPart, $block, $secondPart);

// Décode les balises HTML qui étaient encodées avec htmlentities()
for ($i = 0, $imax = count($page); $i < $imax; $i++) {
    $page[$i] = html_entity_decode($page[$i]);
}

// Mets à jour le code HTML du fichier test.tpl
// stock dans $this->page
$this->page = implode("\n", $page);
}
}
```

## Méthode parse()

### méthode parse()

```
function parse() {  
    $this->constantReplace();  
  
    echo $this->page;  
}
```

## 4 - Méthode(s) supplémentaire(s)

Je vous présente ci-dessous quelques méthodes supplémentaires qui sont utiles dans certains cas (il n'y en a qu'une pour le moment).

### 4.1 - Méthode includeFile()

#### Cette méthode a pour but de

- Gérer l'inclusion de fichier.

#### méthode includeFile()

```
function includeFile ($file) {
    // Enclenche la temporisation de sortie
    ob_start();

    include $file;

    // Enregistre le contenu du tampon de sortie
    $buffer = ob_get_contents();

    // Efface le contenu du tampon de sortie
    ob_clean();

    // Retourne les données enregistrées
    return $buffer;
}
```

Toutefois, pour que cette méthode prenne effet, il vous faut ajouter deux fois la même ligne à la méthode constantReplace().

#### Code à remplacer

```
// Remplace {CONSTANTE} par les données correspondantes
// et met à jour le code HTML du fichier test.tpl
// stocké dans $this->page

// Si $data est un fichier
$data = (file_exists($data)) ? $this->includeFile($data) : $data;
$this->page = preg_replace(`{'.$constant.'}`', $data, $this->page);

// Un peu plus loin dans le code

// Remplacement des constantes par leurs données
foreach($info[$i] as $constant => $data) {
    $data = (file_exists($data)) ? $this->includeFile($data) : $data;
    $blockTag = preg_replace(`{'.$type.''.$constant.'}`', $data, $blockTag);
}
```

## 5 - Conclusion

### 5.1 - Épilogue

Un système de template simple n'est pas très difficile à faire. Il suffit juste de bien comprendre comment est fait le tableau \$infoTpl.

### 5.2 - Liens

 **Site : Comparatif des systèmes de template pour PHP**

 **Forum : Quel système de template utilisez-vous ?**

 **Livre : Smarty - PHP Template Programming and Applications, critique par Pierre-Baptiste Naigeon**

## 6 - Annexe

### 6.1 - Structure de la classe (code PHP)

template.php

```
<?php
class Template {
    private $page;           // Code source HTML de la page - fichier.tpl
    private $infoTpl = array(); // Tableau des constantes => données

    /**
     * Vérifie l'existence du fichier
     */
    function __construct($file) {
    }

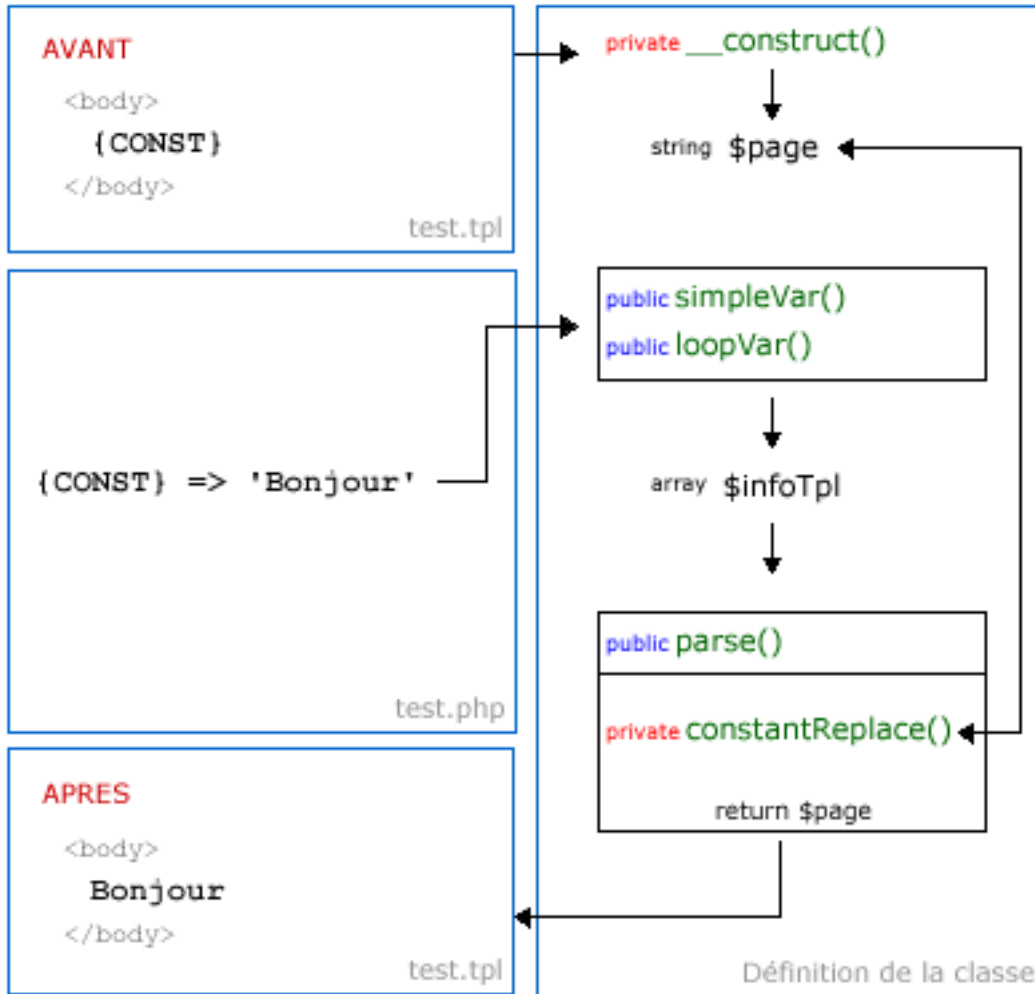
    /**
     * Enregistre les constantes dans $infoTpl
     * infoTpl[.][constant] = data;
     */
    function simpleVar($varArray = array()) {
    }

    /**
     * Enregistre les constantes dans $infoTpl
     * infoTpl[type][lastID][constant] = data;
     *
     * - type      = nom du bloc contenant la boucle
     * - lastID    = ID du tableau où se trouve le script
     */
    function loopVar($type, $varArray = array()) {
    }

    /**
     * Remplace les constantes par leurs données
     */
    function constantReplace() {
    }

    /**
     * Retourne le code HTML parser
     */
    function parse() {
    }
}
?>
```

### 6.2 - Structure avancée de la classe



a-pellegrini.developpez.com

Schéma - structure avancée de la classe

**Explication du schéma :**

- 1 \_\_construct() se lance automatiquement à l'instanciation de la classe;
- 2 Teste si le fichier passé en argument existe et est ouvert en lecture;
- 3 Mise du code source HTML du fichier template dans \$page;
- 4 Récupération des constantes/données de test.php;
- 5 Traitement avec simpleVar() ou loopVar() selon les besoins;
- 6 Mise des constantes/données traitées dans \$infoTpl;
- 7 Parse le code avec parse();
- 8 parse() fait appel à constantReplace();
- 9 constantReplace() remplace les constantes par leurs données.

**6.3 - Structure du tableau \$InfoTpl**

Le tableau \$infoTpl sera de cette forme : *Note : le texte entre accolades représente le nom des variables que j'utilise dans mon code*

```
$infoTpl
```

```
$infoTpl
{
  $infoTpl
  Array
  (
    {
      $type
      [.] => Array
      (
        {
          $info[$i]
          [0] => Array
          (
            {
              $constant => $data
              [WELCOME_MSG] => Bonjour !!
            }
          )
          [1] => Array
          (
            [GOODBYE] => Aurevoir !!
          )
        }
      )
    }
    [country] => Array
    (
      [0] => Array
      (
        [ID] => BE
        [COUNTRY] => Belgique
      )
      [1] => Array
      (
        [ID] => FR
        [COUNTRY] => France
      )
      [2] => Array
      (
        [ID] => ITA
        [COUNTRY] => Italie
      )
    )
  )
}
```

