

Spreadsheet_Excel_Writer en PHP

par Ernaelsten Gérard ([MaitrePylos](#))

Date de publication :

Dernière mise à jour :

Une des demandes les plus fréquentes dans les développements PHP, c'est la génération de fichiers Excel. Dans cet article nous allons générer des fichiers Excel sans passer par la méthode des CSV, de **COM**, ni même de Microsoft Excel ou OpenOffice.org. Tout cela en passant par un fork d'une librairie Perl.

- I - Avant-Propos
- II - Les librairies PHP
 - II-A - php_writeexcel de Johann Hanne
 - II-B - Spreadsheet_Excel_Writer de Xavier Noguier
 - II-C - Besoin complémentaire
- III - Notre premier fichier
 - III-A - La classe de base
 - III-A-1 - La structure
 - III-B - Notre premier fichier Excel
- IV - Feuilles (sheet)
- V - Prenons soin de notre cellule
 - V-A - Array d'options
 - V-B - MaitrePylos à l'envers
 - V-C - Les formules
- VI - Zend Framework
- VII - Exemples
- VIII - Conclusion
- IX - Remerciements

I - Avant-Propos

Dans le monde Perl, il existe une librairie qui permet la génération de fichiers Excel, cette librairie se nomme **Spreadsheet::WriteExcel**, elle fut créée par **John McNamara**. Cette librairie à été adaptée dans le monde PHP, 2 fois. En effet il existe 2 fois la même librairie écrite un peu différemment.

Vous trouverez les sources et fichiers exemples de cet article [ici](#)

II - Les librairies PHP

II-A - php_writeexcel de Johann Hanne

Cette librairie est donc un portage de la classe Perl, vous pouvez la télécharger [ici](#). La dernière mise à jour date d'avril 2002. Cette classe est déjà relativement complète, elle s'inspire directement de la classe Perl, jusqu'à la convention de nommage. Pour la documentation Johann Hanne renvoie à la doc officielle de Perl (en anglais uniquement). Malheureusement toutes les méthodes ne sont pas implémentées et il faut se contenter des exemples mis sur son site pour s'inspirer et générer un fichier Excel.

II-B - Spreadsheet_Excel_Writer de Xavier Noguer

Cette librairie est également un portage de la classe Perl, vous pouvez la télécharger [ici](#). La dernière version bêta date de septembre 2006, celle-ci semble plus complète. Une documentation multilingue se trouve sur le site de **PEAR**. La différence se joue surtout au niveau de la convention de nommage qui ne reprend pas de Perl, en fait elle supprime l'underscore (_), dans le nom de la méthode. La documentation ne fait pas référence à toutes les méthodes, en effet dans le package Perl on trouve de nombreux exemples, que l'on peut s'amuser à faire en PHP (vous en trouverez d'ailleurs pas mal dans les sources), et en réalisant quelques portages on se rend compte que certaines méthodes fonctionnent, alors qu'elles ne sont pas dans la documentation (la méthode `sheet()` notamment).

II-C - Besoin complémentaire

Pour réaliser nos fichiers Excel, nous aurons également besoin de la librairie OLE de **PEAR**. Tandis que Johann Hanne l'inclut directement dans sa librairie, pour utiliser la librairie de Xavier Noguer il faudra la télécharger sur le site de **PEAR**

III - Notre premier fichier

J'ai une préférence pour le travail de Xavier Noguer, et c'est donc à partir de ses classes que je vais développer ce petit article.


III-A - La classe de base

III-A-1 - La structure

Organisation de la librairie

Un des avantages de la librairie de Xavier Noguer, est un projet PEAR, et par conséquent il utilise une convention de nommage strict, qui est également repris par le projet Zend Framework.

Dés lors l'intégration de cette librairie dans un projet Zend Framework ne pose aucun problème.

 *Le nom de classe choisi par l'auteur de la classe ne respecte pas totalement les conventions établies par PEAR. La classe **Spreadsheet_Excel_Writer** devrait ainsi s'appeler par exemple **Spreadsheet_Excel_Writer_Workbook_Main***

Classe de base

```
class Spreadsheet_Excel_Writer extends Spreadsheet_Excel_Writer_Workbook
{
    /**
     * The constructor. It just creates a Workbook
     *
     * @param string $filename The optional filename for the Workbook, $filename create temporary
     file.
     * @return Spreadsheet_Excel_Writer_Workbook The Workbook created
     */
    function Spreadsheet_Excel_Writer($filename='tmp.xls')
    {
        $this->Spreadsheet_Excel_Writer_Workbook($filename);
    }

    /**
     * Send HTTP headers for the Excel file.
     *
     * @param string $filename The filename to use for HTTP headers
     * @access public
     */
    function send($filename)
    {
        header("Content-type: application/vnd.ms-excel");
        header("Content-Disposition: attachment; filename=\"$filename\"");
        header("Expires: 0");
        header("Cache-Control: must-revalidate, post-check=0,pre-check=0");
        header("Pragma: public");
    }

    /**
     * Utility function for writing formulas
     * Converts a cell's coordinates to the A1 format.
     *
     * @access public
     * @static
     */
}
```

Classe de base

```
* @param integer $row Row for the cell to convert (0-indexed).
* @param integer $col Column for the cell to convert (0-indexed).
* @return string The cell identifier in A1 format
*/
function rowcolToCell($row, $col)
{
    if ($col > 255) { //maximum column value exceeded
        return new PEAR_Error("Maximum column value exceeded: $col");
    }

    $int = (int)($col / 26);
    $frac = $col % 26;
    $chr1 = '';

    if ($int > 0) {
        $chr1 = chr(ord('A') + $int - 1);
    }

    $chr2 = chr(ord('A') + $frac);
    $row++;

    return $chr1 . $chr2 . $row;
}
}
```

Cette classe comme expliqué sur la documentation de PEAR est la ligne d'entrée, pour créer notre fichier Excel.

Malheureusement elle ne m'a pas donné que des satisfactions, surtout intégrée dans un projet Zend_Framework. J'ai donc quelque peu modifié cette classe en m'inspirant de la méthode de Johann Hanne, qui consiste à créer un fichier temporaire et à le lire.

Classe modifiée

```
class Spreadsheet_Excel_Writer extends Spreadsheet_Excel_Writer_Workbook
{
    private $_filename;
    /**
     * The constructor. It just creates a Workbook
     *
     * @param string $filename The optional filename for the Workbook, $filename create temporary
     file.
     * @return Spreadsheet_Excel_Writer_Workbook The Workbook created
     */
    function Spreadsheet_Excel_Writer($filename = 'test.xls')
    {
        $this->Spreadsheet_Excel_Writer_Workbook($filename);
    }

    /**
     * Send HTTP headers for the Excel file.
     *
     * @param string $filename The filename to use for HTTP headers
     * @access public
     */
    function send($filename='fichier.xls')
    {
        header("Content-type: application/vnd.ms-excel");
        header("Content-Disposition: attachment; filename=\"$filename\"");
        header("Expires: 0");
        header("Cache-Control: must-revalidate, post-check=0,pre-check=0");
        header("Pragma: public");
    }
}
```

Classe modifiée

```
/**
 * Utility function for writing formulas
 * Converts a cell's coordinates to the A1 format.
 *
 * @access public
 * @static
 * @param integer $row Row for the cell to convert (0-indexed).
 * @param integer $col Column for the cell to convert (0-indexed).
 * @return string The cell identifier in A1 format
 */
function rowcolToCell($row, $col)
{
    if ($col > 255) { //maximum column value exceeded
        return new PEAR_Error("Maximum column value exceeded: $col");
    }

    $int = (int)($col / 26);
    $frac = $col % 26;
    $chr1 = '';

    if ($int > 0) {
        $chr1 = chr(ord('A') + $int - 1);
    }

    $chr2 = chr(ord('A') + $frac);
    $row++;

    return $chr1 . $chr2 . $row;
}

/**
 * Read file create tempory file
 * @access public
 *
 */
function sendFile()
{
    readfile($this->_filename);
    unlink($this->_filename);
}
}
```

Concrètement, qu'est ce que j'ai changé? :

J'ai ajouté une méthode pour la lecture et la destruction du fichier temporaire

```
function sendFile()
{
    readfile($this->_filename);
    unlink($this->_filename);
}
```

III-B - Notre premier fichier Excel

Exemple ici

```
set_time_limit(300);

require_once 'Spreadsheet/Excel/Writer.php';
```

```
$workbook = new Spreadsheet_Excel_Writer();
$workbook->setTempDir('./tempdoc');
$workbook->send('base.xls');
$worksheet = $workbook->addWorksheet();

$worksheet->write(1,2,'toto');

$workbook->close();
$workbook->sendFile();
```

Examinons ce fichier ligne à ligne.

```
set_time_limit(300);
```

Set_time_limit(), permet de dépasser le temps d'attente qui généralement est limité à 30 secondes, ce qui peut-être nécessaire pour le fichier **suivant** par exemple, et qu'on augmente la boucle dans de plus grandes proportions. Malheureusement, pour des raisons de sécurité la fonction est désactivée chez Developpez.com, je vais donc la supprimer de mes exemples. Heureusement, très peu de scripts ont besoin de Set_time_limit().

```
require_once 'Spreadsheet/Excel/Writer.php';
```

On inclut le fichier de la classe Writer, pour pouvoir instancier un objet Spreadsheet_Excel_Writer().

Cette ligne est inutile dans un projet Zend Framework si vous utilisez Zend_Loader::registerAutoload().

```
$workbook = new Spreadsheet_Excel_Writer();
```

Nous créons une instance de notre classe, nous pouvons si nous le voulons lui passer en paramètre le nom d'un fichier temporaire.

```
$workbook->setTempDir('./tempdoc');
```

Définit le dossier temporaire à utiliser pour le fichier OLE. Utilisez cette méthode si vous n'avez pas le droit d'écrire dans le dossier temporaire par défaut. (1)

```
$workbook->send('base.xls');
```

Nous envoyons les entêtes qui permettent de dire aux navigateurs que nous voulons un fichier Excel, ici également vous pouvez passer en paramètre le nom que votre fichier portera.

Si nous arrêtons notre script à ce stade nous n'aurions qu'une application Excel, qui s'ouvrirait complètement vide.

```
$worksheet = $workbook->addWorksheet();
```

Ici, nous créons notre première Feuille(sheet), si vous désirez donner un nom à l'onglet de la feuille, vous pouvez lui passer ce nom en paramètre addWorksheet('Feuille1').

Pour plus de détails sur les Feuilles, reportez-vous au chapitre 3.

```
$worksheet->write(1,2,'toto');
```

Ici nous écrivons dans une cellule.

la méthode write prends trois paramètres minimum :

- La ligne du document Excel (cela commence à zéro)
- La colonne du document Excel (cela commence à zéro)
- Ce que l'on veut mettre dans la cellule (texte, formule etc..)

Dans le cas ici, nous afficherons donc dans la cellule 2C, le mot : 'toto' **Voici** de quoi vous faciliter la navigation dans les cellules.

Pour plus de détails sur les cellules, reportez-vous au chapitre 4.

```
$workbook->close();
```

Nous fermons l'objet

```
$workbook->sendFile();
```

Nous envoyons la lecture de notre fichier.

Nous venons de réaliser notre premier fichier Excel.

IV - Feuilles (sheet)

Dans le fichier de base, nous appelons une méthode "addWorksheet()", afin d'avoir une feuille Excel (espace de travail).

Dans cette feuille nous pouvons passer certains paramètres de configuration via des méthodes setter.

Voici les paramètres :

- setPortrait : Définit l'orientation de la page en tant que portrait
- setLandscape : Définit l'orientation de la page en tant que paysage
- setPaper : Définit le type de papier. Ex. 1 = US Letter, 9 = A4
- setHeader : Définit l'en-tête de la page et, optionnellement, la marge
- setFooter : Définit le pied de page et, optionnellement, la marge
- centerHorizontally : Centre la page horizontalement
- centerVertically : Centre la page verticalement
- setMargins : Définit toutes les marges de la page à la même valeur, en pouces
- setMargins_LR : Définit les marges gauches et droites à la même valeur, en pouces
- setMargins_TB : Définit les marges supérieures et inférieures à la même valeur, en pouces
- setMarginLeft : Définit la marge gauche, en pouces
- setMarginRight : Définit la marge droite, en pouces
- setMarginTop : Définit la marge supérieure en pouces
- setMarginBottom : Définit la marge inférieure, en pouces

```
<?php

//set_time_limit(300);

require_once 'Spreadsheet/Excel/Writer.php';

$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('woorksheet.xls');
$worksheet = $workbook->addWorksheet('TestDeWoorkSheet');
$worksheet->setPaper(9);//Définit une page A4
$worksheet->setHeader('Mon beau fichier Excel');//Définit un entête ,faites appercu pour voir l'entête
$worksheet->setLandscape ();//Définit une orientation Paysage.

$worksheet->write(0,0,utf8_decode('première cellule d\'une page A4,paysage '));
$worksheet->write(31,10,utf8_decode('dernière cellule d\'une page A4,paysage '));

$workbook->close();
$workbook->sendFile();

?>
```

Nous créons donc une feuille.

```
$worksheet = $workbook->addWorksheet('TestDeWoorkSheet');
```

Et nous passons par les méthodes setter, nos paramètres

```
$worksheet->setPaper(9); //Définit une page A4
$worksheet->setHeader('Mon beau fichier Excel'); //Définit un entête ,faites appercu pour voir
l'entête
$worksheet->setLandscape (); //Définit une orientation Paysage.
```

exemple

V - Prenons soin de notre cellule

Notre librairie ne se limite pas à écrire dans des sheets ou des cellules, nous pouvons y mettre quelques options intéressantes

V-A - Array d'options

```
<?php
require_once "Spreadsheet/Excel/Writer.php";
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('format.xls');
$format = $workbook->addFormat(
    array(
        'Size' => 10, //taille du texte
        'Align' => 'center', //alignement du texte
        'Color' => 'white', //couleur du texte
        'FgColor' => 'magenta')); //couleur du fond de cellule
$worksheet = $workbook->addWorksheet();
$worksheet->write(1, 0, "Bonjour en Magenta !", $format);
$workbook->close();
$workbook->sendFile();
?>
```

Cet exemple provient directement de la doc PEAR, cela va nous créer un fichier Excel avec une seule cellule écrite avec une taille 10 en blanc sous fond magenta. Concrètement, nous créons une nouvelle variable \$format dans laquelle nous appelons la méthode "addformat" et lui passons un tableau de paramètres.

Cette variable sera passer en 4ième argument de la méthode write.

Définition d'une cellule formatée :

```
$format = $workbook->addFormat(
    array(
        'Size' => 10, //taille du texte
        'Align' => 'center', //alignement du texte
        'Color' => 'white', //couleur du texte
        'FgColor' => 'magenta')); //couleur du fond de cellule
```

Nous passons la variable en argument :

```
$worksheet->write(1, 0, "Bonjour en Magenta !", $format);
```

Voici les paramètres possibles.

- Align : Définit l'alignement de la cellule
- VAlign : Définit l'alignement de la cellule
- HAlign : Définit l'alignement de la cellule
- Merge : Cette méthode est un alias de la méthode non-intuitive setAlign('merge')
- Bold : Définit l'intensité de la mise en gras d'un texte

- Bottom : Définit l'épaisseur de la bordure inférieure de la cellule
- Top : Définit l'épaisseur de la bordure supérieure de la cellule
- Left : Définit l'épaisseur de la bordure gauche de la cellule
- Right : Définit l'épaisseur de la bordure droite de la cellule
- Border : Définit les bordures d'une cellule dans un même style
- BorderColor : Définit toutes les bordures de la cellule à une même couleur
- BottomColor : Définit la couleur de la bordure inférieure de la cellule
- TopColor : Définit la couleur de la bordure supérieure de la cellule
- LeftColor : Définit la couleur de la bordure gauche de la cellule
- RightColor : Définit la couleur de la bordure droite de la cellule
- FgColor : Définit la couleur de premier-plan de la cellule
- BgColor : Définit la couleur d'arrière-plan de la cellule
- Color : Définit la couleur pour le contenu de la cellule
- Pattern : Définit les attributs d'une cellule
- Underline : Définit le soulignement du texte
- Italic : Définit le style de la police de caractère en italique
- Size : Définit la taille de la police de caractères
- TextWrap : Définit le retour automatique à la ligne du texte
- TextRotation : Définit l'orientation du texte
- NumFormat : Définit le format numérique
- StrikeOut : Définit la police de caractères comme barrée
- OutLine : Définit le dessin d'une police de caractères
- Shadow : Définit la police de caractères comme ombrée
- Script : Définit le type de script pour le texte
- FontFamily : Définit la famille de police de caractères

Nous pouvons également combiner les formats en créant des arrays et en utilisant `array_merge()`;

exemple [ici](#).

```
<?php
require_once "Spreadsheet/Excel/Writer.php";
$workbook = new Spreadsheet_Excel_Writer();
$workbook->send('format.xls');
$couleur = array(
    Color => 'white', //couleur du texte
    FgColor => 'magenta' //couleur du fond de cellule
);
$taille = array(
    Size => 10, //taille du texte
    Align => 'center' //alignement du texte
);
$Merge = array_merge($couleur,$taille);

$format1 = $workbook->addformat($taille);
$format2 = $workbook->addformat($couleur);
$formatMerge = $workbook->addformat($Merge);
$worksheet = $workbook->addWorksheet();
$worksheet->write(1, 0, "format 1", $format1);
$worksheet->write(2, 0, "format 2", $format2);
```

```
$worksheet->write(3, 0, "format merge ", $formatMerge);  
$workbook->close();  
$workbook->sendFile();  
?>
```

V-B - MaitrePylos à l'envers

Mais le plus simple dans le format de cellule est d'utiliser son setter, en effet chaque propriété peut être appelée séparément et ajoutée à 'addformat()'.

```
<?php  
require_once "Spreadsheet/Excel/Writer.php";  
$workbook = new Spreadsheet_Excel_Writer();  
$workbook->send('format.xls');  
  
$worksheet = $workbook->addWorksheet();  
  
$format= $workbook->addformat();  
$format->setTextRotation(90);  
$worksheet->write(1, 0, "Maitre", $format);  
  
$format1= $workbook->addformat();  
$format1->setTextRotation(270);  
$worksheet->write(2, 0, "Pylos", $format1);  
  
$format2 = $workbook->addformat();  
$format2->setTextRotation(-1);  
$format2->setFgColor(11);  
  
$worksheet->write(2, 2, "MaitrePylos", $format2);  
  
$workbook->close();  
$workbook->sendFile();  
?>
```

exemple

V-C - Les formules

Un fichier Excel sans formule mathématique ne serait pas un fichier Excel.

Il suffit de noter "=SUM(A1,B2)", par exemple pour avoir l'addition.

```
require_once 'Spreadsheet/Excel/Writer.php';  
  
$workbook = new Spreadsheet_Excel_Writer();  
$workbook->send('stats.xls');  
$worksheet = $workbook->addWorksheet();  
# Set the column width for columns 1  
$worksheet->setColumn(0, 0, 20);  
  
# Create a format for the headings  
$format = $workbook->addFormat();  
$format->setBold();
```

```
# Write the sample data
$worksheet->write(0, 0, 'Sample', $format);
$worksheet->write(1, 0, 1);
$worksheet->write(2, 0, 2);
$worksheet->write(3, 0, 3);
$worksheet->write(4, 0, 4);
$worksheet->write(5, 0, 5);
$worksheet->write(6, 0, 6);
$worksheet->write(7, 0, 7);
$worksheet->write(8, 0, 8);

$worksheet->write(0, 1, 'Length', $format);
$worksheet->write(1, 1, 25.4);
$worksheet->write(2, 1, 25.4);
$worksheet->write(3, 1, 24.8);
$worksheet->write(4, 1, 25.0);
$worksheet->write(5, 1, 25.3);
$worksheet->write(6, 1, 24.9);
$worksheet->write(7, 1, 25.2);
$worksheet->write(8, 1, 24.8);

# Write some statistical functions
$worksheet->write(0, 4, 'Count', $format);
$worksheet->write(1, 4, '=COUNT(B2:B9)');

$worksheet->write(0, 5, 'Sum', $format);
$worksheet->write(1, 5, '=SUM(B2:B9)');

$worksheet->write(0, 6, 'Average', $format);
$worksheet->write(1, 6, '=AVERAGE(B2:B9)');

$worksheet->write(0, 7, 'Min', $format);
$worksheet->write(1, 7, '=MIN(B2:B9)');

$worksheet->write(0, 8, 'Max', $format);
$worksheet->write(1, 8, '=MAX(B2:B9)');

$worksheet->write(0, 9, 'Standard Deviation', $format);
$worksheet->write(1, 9, '=STDEV(B2:B9)');

$worksheet->write(0, 10, 'Kurtosis', $format);
$worksheet->write(1, 10, '=KURT(B2:B9)');

$workbook->close();
$workbook->sendFile();
```




Vous remarquerez deux choses :

- 1) *Il faut un '=' pour démarrer la formule.*
- 2) *Le nom des cellules dans **la formule** est de type nominatif A1-B6-K9 etc. et non plus de type array 0,1,2,3...*

VI - Zend Framework

L'intégration de la librairie dans un projet Zend_framework, ne pose aucun problème.

Il faut mettre les classes dans le répertoire /Library de votre projet Zend Framework, et puis de rajouter **Zend_Loader::registerAutoload()** dans le Bootstrap et à partir de là, plus besoin de définir "require_once 'Spreadsheet/Excel/Writer.php'".

 *Une petite subtilité est quand même à noter pour pouvoir générer un fichier Excel dans le framework.*

Vous ne pouvez appeler votre code Excel dans une méthode Action, au risque de générer le code Html de la page appelante, parce que la vue se lance automatiquement

Vous devez donc dans une méthode action, appeler une méthode Excel, qui une fois terminer redirigera vers la méthode action demandeuseouf :)

```
<?php
class NouveauController extends Zend_Controller_Action
{

    function IndexAction()
    {
        #faire différents traitements, puis appeler la méthode pour le fichier Excel
        $this->Excel;
    }

    private function Excel()
    {
        /**
         * On génère le fichier de base
         */
        $workbook = new Spreadsheet_Excel_Writer();
        $workbook->send('base.xls');
        $worksheet = $workbook->addWorksheet();

        $worksheet->write(1,2,'toto');

        $workbook->close();
        $workbook->sendFile();
        /**
         * Je renvoie vers l'action qui m'a appelée.
         */
        $this->_redirect($this->IndexAction());
    }
}
?>
```

Vous pouvez également empêcher la vue de s'afficher automatiquement avec la méthode suivante.

```
$this->_helper->viewRenderer->setNoRender()
```

VII - Exemples

Vous trouverez ci-dessous différentes possibilités de ce qu'il est possible de faire avec Spreadsheet::WriteExcel.

Chess	BigFile	Colors	demo
ArrayMerge()	Rotation	Merge	Merge 2
Panes	Repeat	Exemple Simple	Stock
Text Wrap	Regions	Stats	WoorkSheet

VIII - Conclusion

Nous venons d'avoir un rapide coup d'oeil sur une des possibilités de créer des fichiers Excel en PHP.

Je n'ai pas vu toutes les méthodes de la librairie de Xavier Noguier, je vous laisse le soin d'explorer plus en avant la **documentation** officielle.

J'ai également été confronté à une méthode que je n'ai jamais réussi à implémenter 'insertBitmap()', mais je ne désespère pas :). On peut également noter que j'ai principalement utilisé OpenOffice et Firefox, et que dans le cadre de IE et MS Excel le fichier demande à être enregistré et non pas lu. Il existe d'autres méthodes pour générer des fichiers Excel, cela fera peut-être partie d'un autre article.

Bon travail :)

IX - Remerciements

Je tenais à remercier Frédérique, mon épouse, pour la correction orthographique de cet article.

Ainsi qu'a **Yogui**, pour ses encouragements, ses remarques techniques, orthographique et le temps considérable qu'il m'a consacré.

1 : Pour des raisons de sécurité Developpez.com, n'autorise pas d'écrire dans le répertoire temporaire. Renseignez-vous pour savoir si vous devez utiliser cette méthode.