

Génération de documents PDF avec des scripts PHP utilisant la librairie FPDF

par [Jean-Christophe CORNIC](#)

Date de publication : 15/12/2006

Dernière mise à jour : 08/02/2007

Comment générer un document PDF à partir de son site ? Comment créer automatiquement un dossier PDF visible sur son site ? C'est possible grâce à la classe phpToPDF qui dérive de FPDF. Ce tutoriel vous permettra d'entrer dans le monde merveilleux de la création de documents PDF.

- I - Présentation
 - I-A - Le projet
 - I-B - Installation
 - I-C - La documentation
- II - Premiers pas
 - II-A - Texte seulement
 - II-A-1 - avec la méthode Text
 - II-A-2 - avec la méthode Write
 - II-A-3 - avec la méthode Cell
 - II-A-4 - avec la méthode MultiCell
 - II-B - Image seulement
 - II-C - Sommaire et numéros de page
 - II-D - Enregistrer dans un document
- III - Utilisation avancée
 - III-A - Insérer un graphique
 - III-B - Insérer un tableau
 - III-B-1 - Un tableau simple
 - III-B-2 - Un tableau plus complexe
- IV - Conclusion
- V - Les liens utiles

I - Présentation

I-A - Le projet

L'article présenté utilise comme base la classe **FPDF**.

FPDF est une classe PHP permettant de générer des documents PDF en pur PHP, c'est-à-dire sans utiliser la librairie PDFlib. Le F de **FPDF** signifie Free : vous êtes libres de l'utiliser et de la modifier comme vous le souhaitez. Plusieurs développeurs ont ajouté des classes pour créer à chaque fois de nouvelles choses. La classe de base est **FPDF** et dès que quelqu'un veut ajouter un script, il crée une classe qui en hérite.

Voulant moi-même utiliser plusieurs classes déjà existantes, j'ai copié leurs contenus dans la nouvelle classe **phpToPDF** qui hérite de **FPDF**.

FPDF a d'autres avantages : des méthodes de plus haut niveau.

- Choix des unités, du format des pages et des marges;
- Gestion des en-têtes et pieds de page;
- Saut de page automatique;
- Saut de ligne automatique et justification;
- Images (JPEG et PNG);
- Couleurs;
- Liens;
- Support des polices TrueType et Type1;
- Compression des pages.

FPDF ne nécessite aucune extension (à part zlib pour activer la compression) et fonctionne avec **PHP 4 et PHP 5**.

I-B - Installation

Il suffit de télécharger et de mettre dans le répertoire racine de son site :

- Les sources PHP **fpdf.php** et **phpToPDF.php**;
- Le répertoire "**font/**" qui contient les fonts.

[Télécharger phpToPDF.zip](#)

Quand vous avez installé (copié) les scripts PHP et le répertoire "font" sur votre serveur, vous êtes prêts à générer des documents PDF à partir d'un script PHP.

I-C - La documentation

Les méthodes de base (issues de la classe FPDF) ne sont pas toutes détaillées ici. Vous pouvez voir les descriptions et paramètres sur <http://www.fpdf.org/>

II - Premiers pas

Il faut savoir que les méthodes utilisées écrivent dans une page du document PDF généré. Avant d'utiliser certaines méthodes, il faut se placer dans la page avec la méthode SetXY(x,y) car on ne peut pas passer en paramètre la position désirée. (cf. l'exemple avec la méthode Cell)

Il est obligatoire de mettre la ligne SetFont sinon, la génération ne fonctionne pas...

II-A - Texte seulement

Il y a plusieurs façons d'écrire un texte dans une page.

II-A-1 - avec la méthode Text

Description

Imprime une chaîne de caractères. L'origine est à gauche du premier caractère, sur la ligne de base. Cette méthode permet de positionner précisément une chaîne dans la page, mais il est généralement plus simple d'utiliser Cell(), MultiCell() ou Write() qui sont les méthodes standards pour imprimer du texte.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();
$PDF->SetFont("Arial","B",16);
$PDF->Text(40,10,"Uniquement un texte");
$PDF->Output();
```

- **AddPage:** ajoute une page dans le document;
- **SetFont:** détermine la font utilisée (B pour Bold);
- **Text(float x, float y, string txt):** Dans l'exemple ci-dessus, **Text** écrit "Uniquement un texte" en position (40, 10);
- **Output:** permet d'afficher le document généré dans le navigateur.

Résultat

II-A-2 - avec la méthode Write

Description

Cette méthode imprime du texte à partir de la position courante. Lorsque la marge droite est atteinte (ou que le caractère \n est rencontré), un saut de ligne est effectué et le texte continue à partir de la marge gauche. Au retour de la méthode, la position courante est située juste à la fin du texte. Il est possible de mettre un lien sur le texte.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
$PDF->Write(10, "Ceci est un texte multilignes \nEt voici la deuxième ligne");
$PDF->Output();
```

- **Write(float h, string txt [, mixed link])** Dans l'exemple ci-dessus, **Write** écrit le texte *"Ceci est un texte multilignes \nEt voici la deuxième ligne"* avec un saut de ligne de 10 mm.

Résultat

II-A-3 - avec la méthode Cell

Description

Imprime une cellule (zone rectangulaire) avec éventuellement des bords, un fond et une chaîne de caractères. Le coin supérieur gauche de la cellule correspond à la position courante. Le texte peut être aligné ou centré. Après l'appel, soit la position courante se déplace à droite, soit un retour à la ligne est effectué. Il est possible de mettre un lien sur le texte.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();

//Sélection de la police
$PDF->SetFont('Arial','B',16);

//Décalage de 8 cm à droite
$PDF->Cell(80);

//Texte centré dans une cellule 20*10 mm encadrée et retour à la ligne
$PDF->Cell(20,10,'Titre',1,1,'C');
$PDF->Output();
```

- **Cell(80);** écrit une cellule vide sans bord de 80 mm de large à partir de l'endroit où l'on se trouve, c'est-à-dire par défaut, en position (margeLeft, margeTop). Les marges ont la valeur 10 mm par défaut, pour les changer, utiliser SetMargins(); L'appel **setXY(10, 90);** aurait été similaire;
- **Cell(float w [, float h [, string txt [, mixed border [, int ln [, string align [, int fill [, mixed link]]]]]])** Dans l'exemple ci-dessus, **Cell** écrit une cellule de taille (20,10), contenant le texte 'Titre', avec un bord, retour à la ligne et centré.

Résultat

II-A-4 - avec la méthode MultiCell

Description

Cette méthode permet d'imprimer du texte avec des retours à la ligne. Ceux-ci peuvent être automatiques (dès que le texte atteint le bord droit de la cellule) ou explicites (via le caractère \n). Autant de cellules que nécessaire sont imprimées, les unes en dessous des autres.

Le texte peut être aligné, centré ou justifié. Le bloc de cellules peut être encadré et le fond coloré.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();

//Sélection de la police
$PDF->SetFont('Arial','B',16);
```

```
$PDF->MultiCell(0, 10, "Ceci est un texte multilignes centré avec un bord\nEt voici la deuxième ligne", 1, "C", 0);
$PDF->Output();
```

- **MultiCell(float w, float h, string txt [, mixed border [, string align [, int fill]]])**; Dans l'exemple ci-dessus, **MultiCell** écrit une cellule de taille (0, 10), contenant le texte entre guillemets, avec un bord, centré et sans remplissage de la cellule.

Résultat

II-B - Image seulement

Description

C'est la méthode **Image(string file, float x, float y [, float w [, float h [, string type [, mixed link]]]])** qui est utilisée. Elle place une image dans la page. Le coin supérieur gauche doit être spécifié.

- largeur et hauteur explicites (exprimées dans l'unité utilisateur);
- Une dimension explicite, l'autre étant calculée automatiquement afin de respecter les proportions de l'image originale;
- Aucune dimension explicite, auquel cas l'image est dimensionnée en 72 dpi.

Les formats supportés sont le JPEG et le PNG.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
$PDF->Image("./images/kitlogo.jpg", 50, 100);
$PDF->Output();
```

- **Image(string file, float x, float y [, float w [, float h [, string type [, mixed link]]]])** Dans l'exemple ci-dessus, **Image** met l'image *./images/kitlogo.jpg* en position (50, 100)

Résultat

II-C - Sommaire et numéros de page

Description

Il est aussi possible de numéroter ses pages et de générer automatiquement un sommaire. Il suffit de spécifier la page à partir de laquelle vous voulez commencer la numérotation et celle à partir de laquelle vous voulez arrêter la numérotation. Ensuite, pour chaque item du sommaire, vous devez ajouter son nom. Pour finir, il faut ajouter le sommaire sur la page désirée.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->SetFont('Times','',12);
$PDF->AddPage();
$PDF->Cell(0,5,'Page de garde',0,1,'C');
$PDF->AddPage();
```

```
// A partir de cette page, la numérotation commence...
```

```
$PDF->startPageNums();
$PDF->Cell(0,5,'TOC1',0,1,'L');
```

```
// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC1', 0);
$PDF->Cell(0,5,'TOC1.1',0,1,'L');
```

```
// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC1.1', 1);
$PDF->AddPage();
$PDF->Cell(0,5,'TOC2',0,1,'L');
```

```
// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC2', 0);
$PDF->AddPage();
for($i=3;$i<=80;$i++){
    $PDF->Cell(0,5,'TOC'.$i,0,1,'L');
```

```
    // On ajoute un item au sommaire
    $PDF->TOC_Entry('TOC'.$i, 0);
}
```

```
// On arrête ici la numérotation
$PDF->stopPageNums();
$PDF->AddPage();
$PDF->Cell(0,5,'Page non numérotée',0,1,'L');
```

```
//Génère et insère le sommaire en page 2
$PDF->insertTOC(2);
$PDF->Output();
```

- **startPageNums()** Cette méthode commence la numérotation des pages à partir de la page courante;
- **TOC_Entry('titre', 0);** Cette méthode ajoute l'entrée '*titre*' au sommaire;
- **stopPageNums()** Cette méthode termine la numérotation sur la page courante;
- **insertTOC(2)** Cette méthode génère le sommaire en page 2 du document.

Résultat

II-D - Enregistrer dans un document

Description

Il est intéressant d'afficher directement le document généré avec la méthode **Output()** mais il est d'autant plus intéressant d'enregistrer le document généré sur le serveur et de l'afficher à n'importe quel moment et n'importe où dans son site. Voici la procédure...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->SetFont('Times','',12);
$PDF->AddPage();
// on écrit ce que l'on veut dans le document PDF...

// enregistre le document test.PDF dans le répertoire local du serveur.
$PDF->Output("test.PDF", "F");

// affiche le document test.PDF dans une iframe.
echo '
    <iframe src="test.PDF" width="100%" height="100%">
    [Your browser does <em>not</em> support <code>iframe</code>,
    or has been configured not to display inline frames.
    You can access <a href="./test.PDF">the document</a>
    via a link though.]</iframe>
's;
```

- **Output("test.PDF, "F")** Cette méthode enregistre le document généré dans le document *./test.PDF* du serveur;
- **le bloc echo '...'** Ce bloc permet d'afficher (si possible) le document *./test.PDF* dans une iframe.

III - Utilisation avancée

III-A - Insérer un graphique

Nous allons voir ici comment dessiner des droites en couleur dans un repère orthonormé avec un titre, une légende, l'affichage des abscisses et des ordonnées.

1) D'abord et comme pour tous les exemples, le code minimum pour débiter son script PHP...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
```

2) Ensuite, la création des droites que nous voulons afficher...

Elles ne sont pas définies mathématiquement ($y=ax+b$) mais elles sont définies par deux ordonnées, y_1 et y_2 . Ce graphique peut donc nous montrer une progression avec ou sans intervalle.

```
$droite1 = array(0, 100, array(255,0,0), "droite 1");
$droite2 = array(50, 25, array(0,255,0), "droite 2");
$droite3 = array(12, 45, array(0,0,255), "");

$droites = array($droite1, $droite2, $droite3);
```

- **array(0, 100, ...);** 0 et 100 sont les ordonnées y_1 et y_2 ;
- **array(..., array(255,0,0),...);** array(255,0,0) définit la couleur de la droite;
- **array(... "droite 1");** "droite 1" définit le nom de la droite dans la légende. Si cet argument est "", il n'y aura pas de légende pour cette droite;
- **\$droites = array(\$droite1, \$droite2, \$droite3);** C'est cette variable qui sera envoyée à la méthode **setRepere**.

3) Enfin, l'appel de la méthode setRepere pour générer le graphique et l'affichage du document PDF.

```
$PDF->setRepere("Titre du graphique", 30, 80, 100, 60, array("Evolution du PIB de la Creuse en 1956"), array(0, 100, 5), $droites);
$PDF->Output();
```

- **\$titre** C'est le titre du graphique (centré en haut du graphique);
- **\$posX et \$posY** Ce sont les coordonnées du coin en haut à gauche du graphique;
- **\$sizeX et \$sizeY** Ce sont les dimensions du graphique;
- **\$datasX** C'est un tableau à un ou deux éléments. S'il n'y a qu'un élément, c'est une légende de l'axe des abscisses, sinon, le premier élément correspond au départ de la droite et le deuxième correspond à la fin de la droite (ex: array("année N-1", "année N"));
- **\$datasY** C'est un tableau dont les deux premiers éléments sont les min et max de l'ordonnée. Le troisième argument est le nombre d'intervalles entre les min et max;
- **\$droites** C'est le tableau qui contient les droites. (cf. 2)

Résultat

III-B - Insérer un tableau

Les méthodes utilisées pour cela sont inspirées d'un code de www.fpdf.org.

Nous allons voir ici comment définir un tableau composé d'un header et d'un contenu.

III-B-1 - Un tableau simple

1) D'abord et comme pour tous les exemples, le code minimum pour débiter son script PHP...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
```

2) Ensuite, la création des variables utilisées pour la génération du tableau.

```
// Définition des propriétés du tableau.
$proprietesTableau = array(
    'TB_ALIGN' => 'L',
    'L_MARGIN' => 15,
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => '0.3',
);

// Définition des propriétés du header du tableau.
$proprieteHeader = array(
    'T_COLOR' => array(150,10,10),
    'T_SIZE' => 12,
    'T_FONT' => 'Arial',
    'T_ALIGN' => 'C',
    'V_ALIGN' => 'T',
    'T_TYPE' => 'B',
    'LN_SIZE' => 7,
    'BG_COLOR_COLO' => array(170, 240, 230),
    'BG_COLOR' => array(170, 240, 230),
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => 0.2,
    'BRD_TYPE' => '1',
    'BRD_TYPE_NEW_PAGE' => '',
);

// Contenu du header du tableau.
$contenuHeader = array(
    50, 50, 50,
    "Titre de la première colonne", "année N-1", "année N",
);

// Définition des propriétés du reste du contenu du tableau.
$proprieteContenu = array(
    'T_COLOR' => array(0,0,0),
    'T_SIZE' => 10,
    'T_FONT' => 'Arial',
    'T_ALIGN_COLO' => 'L',
    'T_ALIGN' => 'R',
    'V_ALIGN' => 'M',
    'T_TYPE' => '',
    'LN_SIZE' => 6,
    'BG_COLOR_COLO' => array(245, 245, 150),
    'BG_COLOR' => array(255,255,255),
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => 0.1,
```

```

        'BRD_TYPE' => '1',
        'BRD_TYPE_NEW_PAGE' => '',
    );

    // Contenu du tableau.
    $contenuTableau = array(
        "champ 1", 1, 2,
        "champ 2", 3, 4,
        "champ 3", 5, 6,
        "champ 4", 7, 8,
    );

```

- **\$proprietesTableau** Cette variable contient les propriétés du tableau, l'alignement, la marge, la couleur et l'épaisseur de bord;
- **\$proprieteHeader** Cette variable contient les propriétés du header, couleurs, taille, font, alignement du texte ; couleur et taille des bordures;
- **\$contenuHeader** Cette variable contient les largeurs des colonnes et leurs contenus;
- **\$proprieteContenu** Du même type que \$proprieteHeader, cette variable contient les propriétés du reste du tableau (elles peuvent différer...);
- **\$contenuTableau** Cette variable contient le contenu du tableau, pour X colonnes et Y lignes, il faut X*Y items dans ce tableau.

3) Enfin, on génère le tableau et on affiche le résultat.

```

// D'abord le PDF, puis les propriétés globales du tableau.
// Ensuite, le header du tableau (propriétés et données) puis le contenu (propriétés et données)
$PDF->drawTableau($PDF, $proprietesTableau, $proprieteHeader, $contenuHeader, $proprieteContenu,
$contenuTableau);

$PDF->Output();

```

Résultat

III-B-2 - Un tableau plus complexe

Les fonctionnalités présentées ici permettent de différencier non seulement le "header" et le contenu du tableau mais aussi chaque cellule du contenu. Il devient possible de faire varier le style du texte de chaque cellule (aligner, souligner, mettre en italique ou en gras).

1) Il faut toujours commencer par le même code c'est-à-dire :

```

include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);

```

2) C'est à la création des variables qu'il y a des petits changements.

Il devient possible d'ajouter des balises pour chaque cellule du contenu du tableau. Ces balises sont écrites entre crochets et sont classiques (I,U et B pour Italic, Underline et Bold ainsi que LCR pour Left, Centered et Right). Par exemple, si je veux écrire dans une cellule du tableau le texte "Salut" centré et souligné, je mettrais dans cette cellule la chaîne de caractère suivante: "[CU]Salut"

Il est aussi possible de fusionner deux cellules. Pour cela, il suffit de mettre dans la première cellule le texte des cellules fusionnées et de mettre dans la deuxième cellule la chaîne de caractères "COLSPAN2". Dans un tableau à

3 colonnes, si je veux fusionner les deux premières cellules de la première ligne, il faut mettre à la ligne désirée:

"mon texte", "COLSPAN2", "le contenu de ma troisième cellule"

```
// Définition des propriétés du tableau.
$proprieteTableau = array(
    'TB_ALIGN' => 'L',
    'L_MARGIN' => 15,
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => '0.3',
);

// Définition des propriétés du header du tableau.
$proprieteHeader = array(
    'T_COLOR' => array(150,10,10),
    'T_SIZE' => 12,
    'T_FONT' => 'Arial',
    'T_ALIGN' => 'C',
    'V_ALIGN' => 'T',
    'T_TYPE' => 'B',
    'LN_SIZE' => 7,
    'BG_COLOR_COL0' => array(170, 240, 230),
    'BG_COLOR' => array(170, 240, 230),
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => 0.2,
    'BRD_TYPE' => '1',
    'BRD_TYPE_NEW_PAGE' => '',
);

// Contenu du header du tableau.
$contenuHeader = array(
    50, 50, 50,
    'Titre de la première colonne', 'année N-1', 'année N',
);

// Définition des propriétés du reste du contenu du tableau.
$proprieteContenu = array(
    'T_COLOR' => array(0,0,0),
    'T_SIZE' => 10,
    'T_FONT' => 'Arial',
    'T_ALIGN_COL0' => 'L',
    'T_ALIGN' => 'R',
    'V_ALIGN' => 'M',
    'T_TYPE' => '',
    'LN_SIZE' => 6,
    'BG_COLOR_COL0' => array(245, 245, 150),
    'BG_COLOR' => array(255,255,255),
    'BRD_COLOR' => array(0,92,177),
    'BRD_SIZE' => 0.1,
    'BRD_TYPE' => '1',
    'BRD_TYPE_NEW_PAGE' => '',
);

// Contenu du tableau.
$contenuTableau = array(
    "mon texte", "COLSPAN2", "le contenu de ma troisième cellule",
    "[CU]Salut 1", 1, 2,
    "champ 2", 3, 4
);
```

Résultat

[Documentation de la méthode drawTableau](#)

IV - Conclusion

Tout d'abord, merci à Yogui et à Julp pour les relectures !!!

La classe phpToPDF est libre d'utilisation mais tenez-moi au courant de vos mises à jour en m'envoyant un mail (jc_cornic@yahoo.fr).

Si vous êtes intéressés par d'autres scripts freewares pour ajouter des fonctionnalités à cette classe, allez donc voir sur www.fpdf.org à la rubrique scripts.

V - Les liens utiles

[EZPDF](#) Un article qui parle d'une autre librairie pour générer du PDF à partir de scripts PHP.

[forum PHP](#) LE forum qui m'aide à progresser en PHP.

<http://www.fpdf.org> Le site d'où vient la base de la librairie phpToPDF.

[DOC](#) La documentation de certaines méthodes de la classe phpToPDF.