
PDF avec PHP

Hugo Etiévant

*Création de documents PDF avec
la classe EZPDF de R&OS Ltd*

version 009c

La bibliothèque EZPDF

L'organisation *R&OS Ltd* a développé une bibliothèque gratuite pour la création à la volée de documents PDF avec le langage PHP.

Cette bibliothèque se compose d'une classe de base : la classe PDF. Cette dernière a récemment été étendue par la classe EZPDF. Ces classes sont livrées avec des polices de caractères disponibles dans le répertoire */fonts* du package. Ce dernier doit être installé sur votre site.

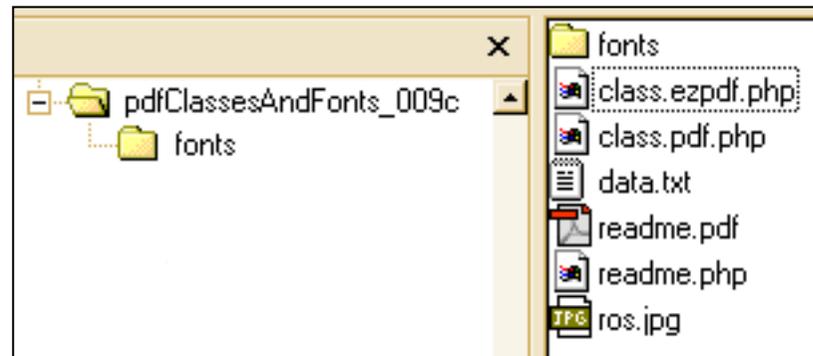
Cette présentation se base sur la version **0.09c** de cette bibliothèque qui est en libre téléchargement sur : <http://www.ros.co.nz/pdf/>.

Ce tutorial est destiné à vous aider dans l'utilisation de la bibliothèque EZPDF. Ce document n'émane pas de R&OS mais s'inspire largement du manuel (en langue anglaise) *readme.pdf* du package. Ce tutorial suppose que le lecteur connaît le langage PHP dont les concepts de classes et d'objets ainsi que celui d'entêtes HTTP.

Dans la suite, les fonctions préfixées par « ez » émanent de la classe EZPDF, les autres de la classe de base PDF.

Installation

Pour installer le package EZPDF,
décompressez l'archive Zip
pdfClassesAndFonts_009c.zip
dans un répertoire de votre site web.



Les fichiers suivants seront alors installés :

class.ezpdf.php : classe EZPDF

class.pdf.php : classe de base PDF

data.txt : manuel en texte brut

readme.php : script générant le manuel en PDF

readme.pdf : manuel en PDF

ros.jpg : logo de *R&OS Ltd*

Et le répertoire **font/** qui contient la liste des polices de caractères utilisables.

Mon premier document PDF avec EZPDF

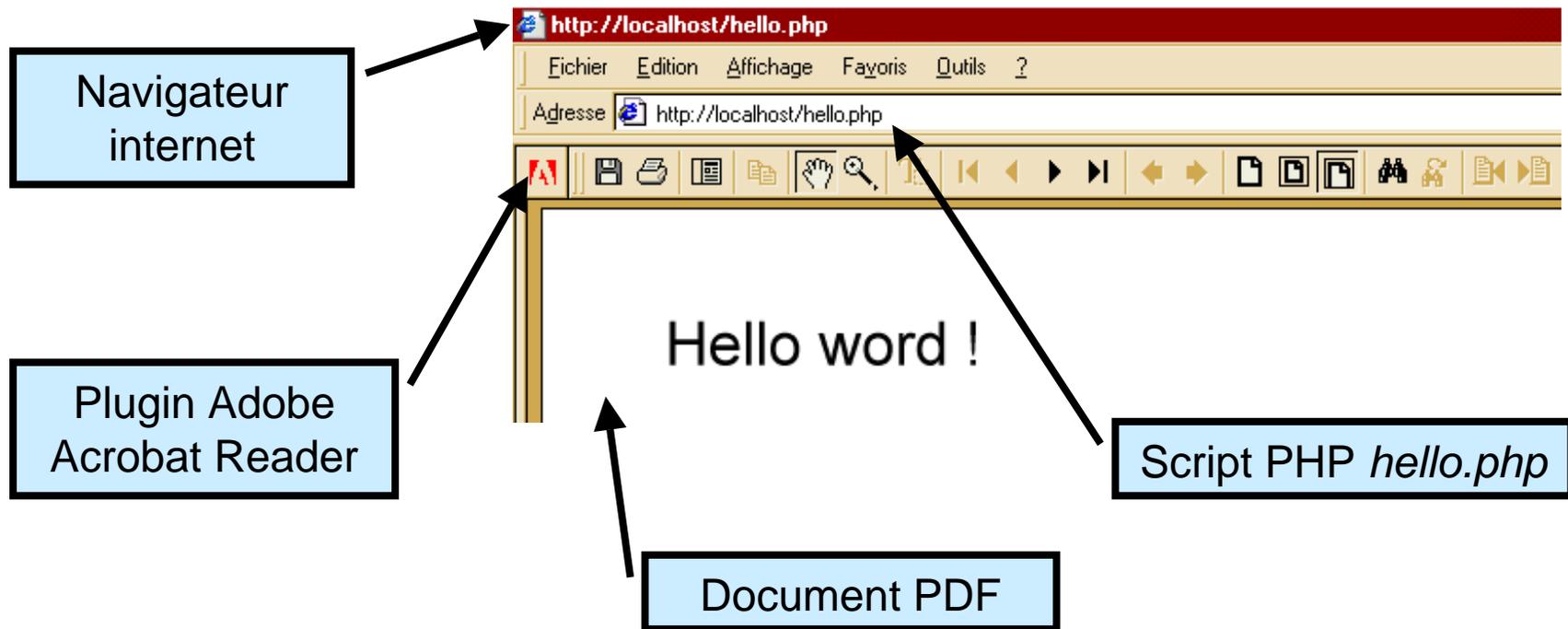
La création d'un document PDF peut se faire de deux manières distinctes : par sa création à la volée et son renvoi au navigateur ou bien en sa création à la volée et son enregistrement sur le serveur. Sera abordé d'abord la première méthode qui suppose l'envoi d'entêtes spécifiques.

```
<?php
include 'class.ezpdf.php'; // inclusion du code de la bibliothèque
$pdf = & new Cezpdf(); // constructeur de la classe EZPDF
$pdf->ezText('Hello word !'); // affichage d'une phrase
$pdf->ezStream(); // envoi du fichier au navigateur
?>
```

Vous avez remarqué que la création d'un document PDF revenait en la manipulation d'un objet **\$pdf** qui est une instance de la classe EZPDF déclarée dans le fichier **class.ezpdf.php**. Comme tout objet, il est créé grâce à un constructeur (ici la méthode **Cezpdf()**) via l'opérateur d'instanciation **new**. L'opérateur **&** assigne à la variable **\$pdf** une référence à l'objet et pas l'objet lui-même.

Ensuite, on utilise la méthode **ezText()** dont on passe en paramètre une chaîne de caractères afin de l'écrire dans le fichier PDF que constitue l'objet **\$pdf**. La méthode **ezStream()** envoie au navigateur un fichier PDF en compilant les informations que contient la variable **\$pdf** et en précisant les entêtes HTTP adéquates.

Résultat dans le navigateur



Le navigateur appelle le plugin *Adobe Acrobat Reader* du navigateur pour afficher le fichier **hello.php** qui retourne un document PDF.

A noter que la fonction **ezText()** n'a aucun effet sur le navigateur, seule la fonction finale **ezStream()** qui génère le fichier PDF provoque l'envoi des entêtes HTTP et des données au format PDF au navigateur.

Constructeur

Cezpdf([string paper='a4'] [, string orientation='portrait'])

Cette méthode est le constructeur de la classe CEZPDF et permet à l'utilisateur de créer rapidement un document PDF. Ses arguments optionnels sont le format du papier (**paper**) et l'orientation (**orientation**).

Les valeurs possibles sont :

paper	orientation
'a4'	'portrait'
'letter'	'landscape' (paysage)

Les valeurs par défauts sont 'a4' et 'portrait'.

Exemples :

\$pdf->Cezpdf();

Création d'un nouveau document avec les paramètres par défaut (A4, portrait).

\$pdf->Cezpdf('a4', 'landscape');

Création d'un document au format A4 et en orientation paysage.

Texte (I)

ezText(string text [, int size] [, array options])

Cette méthode ajoute un bloc de texte dans la page courante aux positions x et y courantes. Le texte sera écrit avec la fonte et les styles en cours ou ceux par défaut. Si le texte est suffisamment grand, il sera écrit sur plusieurs lignes voire, plusieurs pages contigües. L'argument **text** contient la chaîne de caractères à écrire. L'entier **size** est la taille du texte (en unité courante, par défaut en *point*). Et **options** est un tableau associatif dont les clés sont les noms des options et les valeurs la valeur de l'option associée. Le caractère spécial **\n** de saut de ligne est interprété.

Nom de l'option	Valeur
'left'	Nombre, marge de gauche
'right'	Nombre, marge de droite
'aleft'	Nombre, position absolue à gauche
'aright'	Nombre, position absolue à droite
'justification'	Justification du texte : 'left' (à gauche), 'right' (à droite), 'center', 'centre' (au centre), 'full' (collé à gauche et à droite)
'leading' *	Nombre, taille totale prise par une ligne
'spacing' *	Nombre flottant parmi 1, 1.5 et 2 : espace entre chaque ligne.

Texte (II)

Exemple 1 :

```
$pdf->ezText('Hello word');
```

Inclusion dans le document PDF d'un bloc texte contenant le texte 'Hello word'.

Exemple 2 :

```
$pdf->ezText('Hello word', 20);
```

Inclusion du même bloc texte, à la suite du premier mais en forçant sa taille à 20 pt.

Exemple 3 :

```
$tab = array(  
    'justification' => 'center',  
    'spacing' => 1.5 );
```

```
$pdf->ezText($mybook, 20, $tab);
```

Inclusion d'un bloc texte contenant la chaîne de caractères **\$mybook** de taille 20 pt et avec les options suivantes : texte centré et avec des sauts de ligne de taille 1.5 fois la taille d'une ligne normale.

Texte (III)

addText(int x, int y, int size, string text [, int angle=0] [, int adjust=0])

Cette fonction ajoute du texte sur une seule ligne à la position (**x**, **y**). L'origine des axes d'un document PDF est par défaut au coin gauche inférieur de la page. La taille du texte est exprimée en **size** points. Il sera affiché la chaîne de caractères **text**. Il est possible de spécifier un angle de rotation **angle** en degrés dans le sens inversement trigonométrique. On peut aussi spécifier un espacement constant entre chaque lettre du texte : **adjust**. Attention, si le texte est trop long, il sera tronqué : il n'y a pas de passage à la ligne (le caractère spécial **\n** de saut de ligne n'est pas interprété).

Depuis la version 0.06, on peut rajouter des directives de mise en forme des caractères (italique et/ou gras). Ces directives sont similaires aux balises HTML, elles sont en minuscules et ne doivent pas comporter d'espace entre elles et le texte.

Il est conseillé d'utiliser la fonction PHP **htmlspecialchars()** afin de transformer le caractère '<' (respectivement '>') en son code HTML équivalent '<' (respectivement '>') afin d'éviter toute ambiguïté avec une directive.

Directive	Description	Exemple
b	Mise en gras	hello
i	Mise en italique	<i>hello</i>

Texte (IV)

Exemple 1 :

```
$pdf->addText(100, 200, 12, 'Hello word');
```

Ajout du texte 'Hello Word' de taille 12 à la position (100, 200).

Exemple 2 :

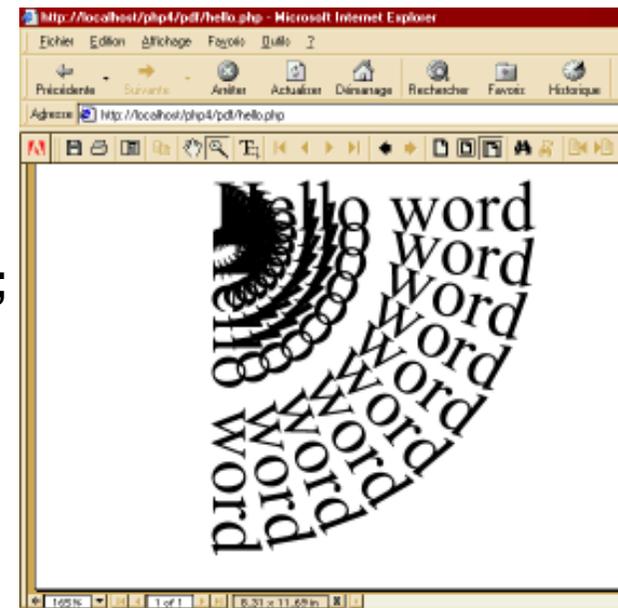
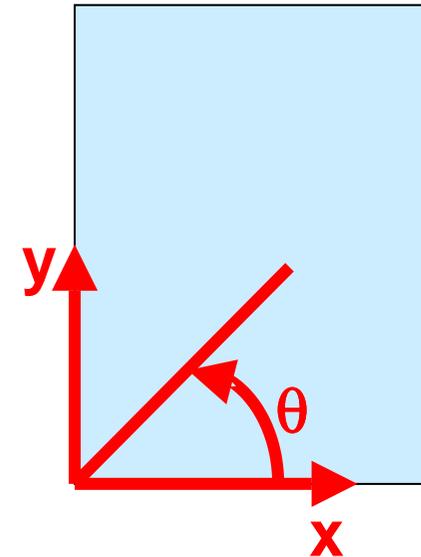
```
$pdf->addText(100, 200, 12, 'Hello word', 0, 10);
```

Idem mais avec un espacement de 10 entre chaque lettre du texte.

Exemple 2 :

```
for($i=0; $i<=90; $i+=10) {  
    $pdf->addText(100, 200, 12, 'Hello word', $i);  
}
```

Idem mais incliné de 45° par rapport à l'horizontale selon le sens trigonométrique.



Texte (V)

selectFont(string fontName [, string|array encoding])

Cette méthode permet de sélectionner la police de caractère à utiliser à partir de l'endroit où elle est appelée. La variable **fontName** contient le nom et chemin (relatif) complet du fichier contenant la police.

Exemples :

```
$pdf->selectFont('./fonts/Times-Roman.afm');
```

```
$pdf->addText(100, 200, 12, 'Hello word');
```

```
$pdf->selectFont('./fonts/Times-Italic.afm');
```

```
$pdf->addText(100, 250, 12, 'Hello word');
```

Tableau des alignements du texte

Valeur	Description
'left'	aligné à gauche
'right'	aligné à droite
'center'	centré
'full'	justifié

Texte (VI)

setFontFamily(string family, array options)

Cette méthode permet d'associer à une police des variantes auxquelles on pourra faire référence grâce aux directives de mise en forme. La chaîne **family** définit la police de base à partir du nom du fichier de la police (qui doit être dans le répertoire **./fonts**, ne pas donner le chemin). Et le tableau **options** définit les polices alternatives en cas de recours aux directives.

Exemple 1 :

```
$pdf->selectFont('./fonts/Times-Roman.afm');  
$pdf->setFontFamily('Times-Roman.afm', array( 'b' => 'Helvetica.afm' ));  
$pdf->ezText('The <b>sky</b> is blue.');
```

Cet exemple affiche une chaîne de caractères. Dans cette chaîne, la directive **** implique la mise en gras du mot **'sky'**. Par défaut, c'est le fichier *Times-Bold.afm* (variante système de *Times-Roman.afm*) qui aurait dû être utilisée. Mais dans les options, on a redéfini le gras en *Helvetica.afm*.

Note : la méthode **selectFont()** doit impérativement avoir été appelée avant la définition de **setFontFamily()** pour que cette dernière soit prise en compte. En effet elle s'applique par association au nom d'une police déjà sélectionnée.

Texte (VII)

On peut combiner les directives afin de mettre en place un jeu complexe de variantes d'une fonte selon le contexte afin d'émuler simplement des styles de paragraphes (MSWord). Exemples : `<i></i>`, ``.

La directive `<u></u>` pour souligner a été ajoutée à la version 009 mais ne fonctionne qu'avec la méthode `ezText()`.

Exemple 2 :

```
$pdf->selectFont('./fonts/Times-Roman.afm');
```

```
$family = 'Times-Roman.afm';
```

```
$options = array(
```

```
    'b' => 'Times-Bold.afm',
```

```
    'i' => 'Times-Italic.afm',
```

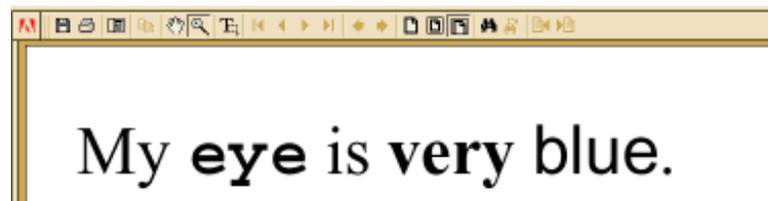
```
    'bi' => 'Helvetica.afm',
```

```
    'bb' => 'Courier-Bold.afm'
```

```
);
```

```
$pdf->setFontFamily($family, $options);
```

```
$pdf->ezText('My <b><b>eye</b></b> is <b>very</b> <b><i>blue</i></b>');
```



Texte (VIII)

string s = addTextWrap(int x, int y, int width, int size, string text [, string justification = 'left'] [, int angle=0])

Proche de **addText()**. Insert un bloc texte à la position (**x**, **y**) de largeur **width** avec le texte **text** dont la taille est **size**. Le texte peut être incliné de **angle** degrés et aligné à gauche ('left'), à droite ('right'), centré ('center') ou justifié ('full'). Si le texte est trop long pour tenir dans le bloc, alors il est tronqué si possible au niveau d'un espace et le reste du texte est retourné.

Exemple :

\$x = 50; // abscisse d'insertion

\$y = 800; // ordonnée d'insertion

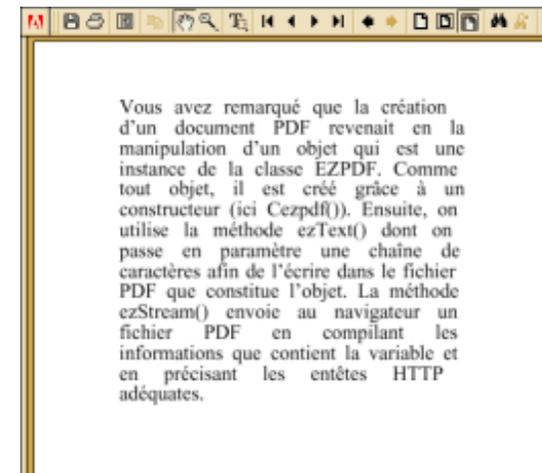
\$size = 12; // taille du texte

\$width = 200; // largeur du bloc texte

while(\$mytext = \$pdf->addTextWrap(\$x, \$y, \$width, \$size, \$mytext)) {

\$y -= 12 ; // décrémentation de l'ordonnée

};



Texte (IX)

int x = getFontHeight(int size)

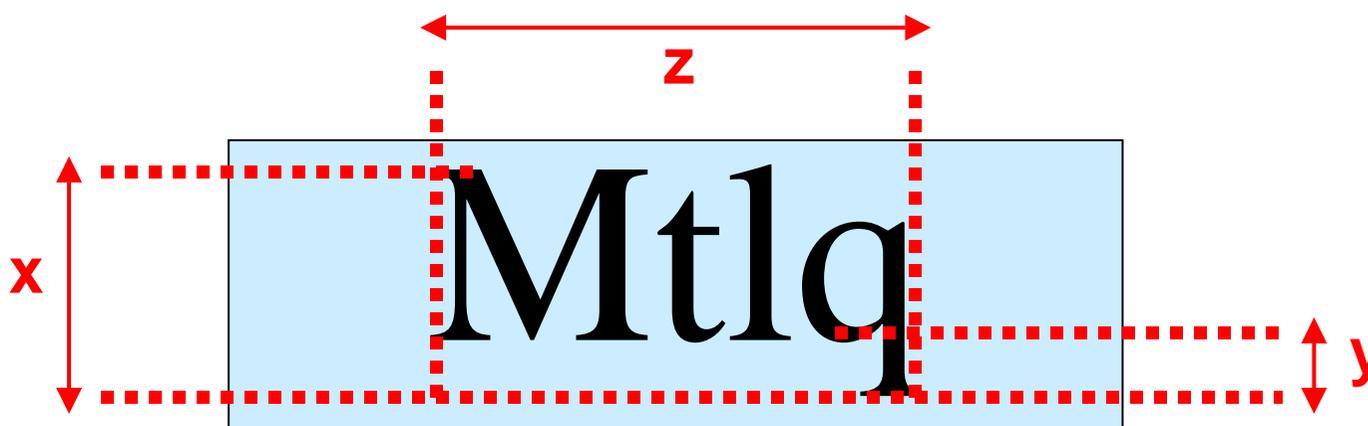
Retourne la hauteur réelle (dans l'unité courante du document PDF) de la fonte en fonction de la taille de caractère **size** choisie. Cette hauteur court du haut de la majuscule jusqu'au bas des lettres à queue.

int y = getFontDecender(int size)

Retourne la hauteur entre la base des lettres et la base de la ligne. Cette hauteur cours du bas des lettres sans queue au bas de celles avec queue.

int z = getTextWidth(int size, int text)

Retourne la longueur du texte **text** dans la taille **size** donnée.



Texte multicolonne

ezColumnsStart([array options])

Permet d'activer le mode multicolonne. Le paramètre **option** est un tableau associatif dont les clés sont : 'num' et 'gap' définissant respectivement le nombre de colonnes et l'espace entre deux colonnes.

ezColumnsStop()

Permet de stopper le mode multicolonne.

Exemple :

```
$pdf->ezColumnsStart(array('num' => 2, 'gap' => 20));
```

```
$pdf->ezText("Le niveau des océans monte d'environ 2.5 millimètres par an depuis 1993. Cette variation a été calculée à partir des mesures effectuées entre 1993 et 2001 par le satellite Topex-Poséidon. ", 12, array('justification' => 'full'));
```

```
$pdf->ezColumnsStop();
```



Liens hypertextes externes (Internet)

addLink(string url, int x0, int y0, int x1, int y1)

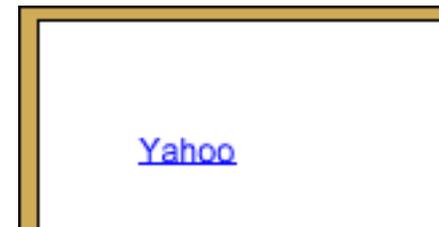
Crée une zone rectangulaire cliquable renvoyant vers une adresse internet **url**. Les coordonnées du rectangle sont données par **(x0,y0)** et **(x1,y1)**. Le rectangle est invisible hormis le curseur de la souris qui change en survolant la zone. C'est pratique pour rendre cliquable une image.

Exemple :

```
$pdf->addLink("http://cyberzoide.developpez.com", 50, 100, 396, 262);  
$pdf->addJpegFromFile("logo.jpg", 50, 100, 346, 162);
```

Cette méthode n'est pas très simple, c'est pourquoi il existe une autre technique qui consiste en l'ajout de code spécial dans le texte lui-même :

```
$pdf->ezText('<c:alink:http://www.yahoo.fr>Yahoo</c:alink>');
```



Liens hypertextes internes au document (I)

Pour créer des liens hypertexte au sein même du document, il faut :

- marquer la position à atteindre grâce à une étiquette
- marquer un ou des mot(s) sur lesquels cliquer pour atteindre la position destination

addDestination(string label, string style [, int a][, int b][,int c])

Cette méthode crée l'ancrage de destination auquel on affecte une étiquette **label** chaîne de caractère qui doit être unique afin d'éviter toute ambiguïté. Les autres paramètres sont identiques à ceux de **openHere()**, ils permettent de forcer le mode d'affichage de l'éditeur PDF...

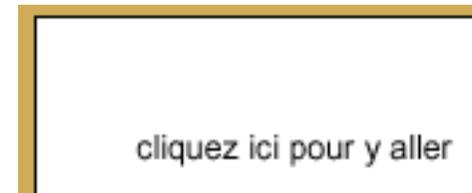
La directive spéciale **<c:ilink:label></c:ilink>** permet de créer le lien hypertexte autour d'un mot vers le point d'ancrage.

Exemple :

```
$pdf->addDestination('label1', 'Fit');
```

...

```
$pdf->ezText('<c:ilink:label1>cliquez ici</c:ilink> pour y aller');
```



Liens hypertextes internes au document (II)

addInternalLink(string label, int x0, int y0, int x1, int y1)

Cette méthode est une alternative à la directive spéciale **ilink** et permet de créer un lien interne sur une image ou une zone du document plutôt que sur du texte.

Exemple :

```
$pdf->addDestination('label1', 'Fit');
```

```
...
```

```
$pdf->filledRectangle(50, 50, 100, 300);
```

```
$pdf->addInternalLink('label1', 50, 50, 150, 350);
```

Mise en page (I)

ezSetMargins(int top, int bottom, int left, int right)

Permet de définir la valeur des marges supérieur, inférieur, gauche et droite. Les valeurs par défaut d'un nouveau document sont 30 points. La méthode **ezText()** respecte ces marges. Par contre les fonctions de base (qui ne portent pas le préfixe 'ez') ne tiennent pas compte des marges.

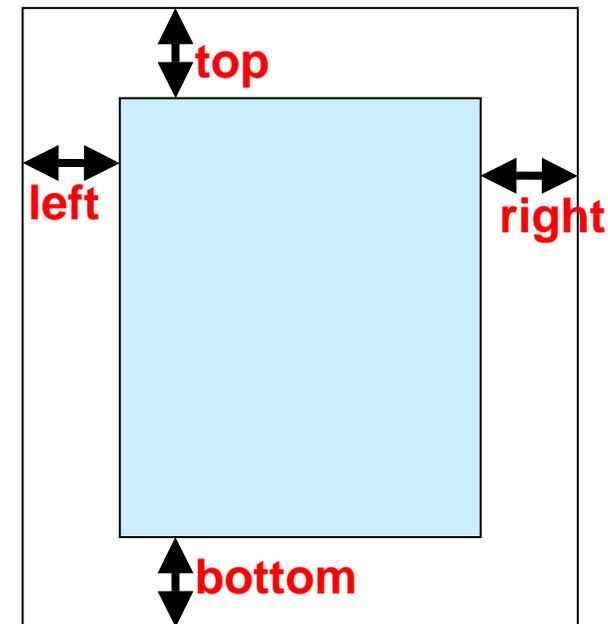
La fonction **ezSetCmMargins()** est identique à la précédente mais utilise les centimètre pour unité.

Exemple :

```
$pdf->ezSetMargins(50,50,100,50);
```

int id = getFirstPageId()

Retourne l'identifiant de la toute première page créée (lors de l'instanciation par l'appel au constructeur).



Mise en page (II)

ezNewPage()

Ajout d'une page au document. Equivalent à « Saut de page » de MSWord.

Exemple :

```
$pdf->ezNewPage();
```

int id = newPage()

Ajout d'une page au document. Retourne l'identifiant numérique de la page afin de permettre l'insertion plus tard d'objets dans cette page en particulier.

Exemple :

```
$id = $pdf->newPage();
```

Quand une nouvelle page est créée, le pointeur de position est automatiquement remis en haut à gauche pour inclusion de texte, image et objet.

Mise en page (III)

ezInsertMode([int status=1, int pageNum=1, int pos='before'])

Permet l'insertion de nouvelles pages au beau milieu du document. Cette fonction doit être appelée deux fois : la première pour activer ce mode (**status=1**) d'insertion, et la seconde pour le désactiver (**status=0**). Les nouvelles pages seront insérées avec **ezNewPage()** relativement à la page numéro **pageNum** : soit avant (**pos='before'**) soit après (**pos='after'**).

Exemple :

```
1 $pdf->ezInsertMode(1, 3, 'before');  
2 $pdf->ezNewPage();  
3 $pdf->ezText($poeme);  
4 $pdf->ezInsertMode(0);  
5 $pdf->ezNewPage();
```

Dans cet exemple, avant la ligne 1, de nombreuses pages ont déjà été créées. A la ligne 1 on active le mode d'insertion pour ajouter les pages suivantes juste avant (**'before'**) la page n°3. La ligne 2 permet de créer une nouvelle page sera donc placée avant celle n°3. La ligne 3 affiche du texte dans cette page. Et la ligne 4 referme ce mode. A la ligne 5 on ajoute une page en fin de document.

Si le numéro de page indiqué par **pageNum** est invalide, alors la ou les nouvelles pages seront insérées en fin de document.

Mise en page (IV)

ezSetY(int y)

Place le pointeur de l'axe des ordonnées à la position spécifiée **y**.
L'origine de l'axe des ordonnées se situe en bas de page.

ezSetDy(int dy [, string mod])

Déplace le pointeur de l'axe des ordonnées de **dy** unités. Si la valeur spécifiée place le pointeur en dehors de la marge inférieure, alors une nouvelle page est créée et le pointeur est automatiquement positionné en haut (en respectant la marge supérieure). Si l'option **mod** est mis à 'makeSpace' et qu'un saut de page est opéré, alors, le déplacement **dy** sera à nouveau effectué depuis la marge supérieure.

Tableaux (I)

ezTable(array data [, array cols] [, string title] [, array options])

Le tableau **data** a pour élément des tableaux associatifs dont les clés sont les nom de colonnes et les éléments, les valeurs. A une ligne du tableau **data** correspond une ligne du tableau du document PDF.

Le tableau associatif **cols** permet de renommer et de changer l'ordre des colonnes du tableau. Les clés de **data** permettant seulement l'ordonnement des éléments dans les bonnes colonnes en les triant si nécessaire.

La chaîne de caractères **title** est le titre du tableau.

Le tableau associatif **options** permet de spécifier des options de mise en forme.

Exemple 1 :

```
$data = array(  
    array('Nom' => 'Pratt', 'Prénom' => 'Hugo'),  
    array('Nom' => 'Verlaine', 'Prénom' => 'Paul'),  
);  
$pdf->ezTable($data);
```

Nom	Prénom
Pratt	Hugo
Verlaine	Paul

Tableaux (II)

Exemple 2 :

```
$data = array(  
    array('Marque' => 'Renault', 'Modèle' => 'Clio', 'Pays' => 'France'),  
    array('Marque' => 'Fiat', 'Modèle' => 'Punto', 'Pays' => 'Italie'),  
);  
$cols = array('Pays' => '<i>Pays d'origine</i>', 'Marque' => 'Entreprise');  
$title = 'Liste des constructeurs';  
$pdf->ezTable($data, $cols, $title);
```

Dans cet exemple, on change l'ordre des colonnes et on renomme les titres de colonnes grâce au tableau associatif **\$cols** dont les clés doivent être les mêmes que celles de **\$data**. On choisit également de ne pas faire apparaître la colonne *Modèle*. On ajoute aussi un titre à notre tableau.

Liste des constructeurs

<i>Pays d'origine</i>	Entreprise
France	Renault
Italie	Fiat

Tableaux (III)

Les clés et valeurs possibles du tableau des **options** sont données ci-après :

Nom de l'option	Valeurs, description
'showLines'	0 (ne trace pas les bordures de ligne) ou 1 (les trace) Par défaut : 1
'showHeadings'	0 (n'affiche pas le titre) ou 1 (l'affiche) Par défaut : 1
'shaded'	0 (ne colore pas le fond d'une ligne sur deux) ou 1 (le fait) Par défaut : 1
'shadeCol'	Tableau des composantes RVB (nombre flottant entre 0 et 1 dans chaque cellule) de la couleur de fond de 'shaded'. Par défaut : array(0.8, 0.8, 0.8)
'shadeCol2'	Idem pour pour l'autre ligne
'fontSize'	Taille du texte. Par défaut : 12.
'textCol'	Couleur du texte. Tableau des composantes RVB.
'titleFontSize'	Taille du titre. Par défaut : 12.
'titleGap'	Espace entre la base du titre et le début du tableau. Il peut être négatif. Par défaut : 5.
'lineCol'	Couleur des traits. Tableau des composantes RVB.

Tableaux (IV)

Nom de l'option	Valeurs, description
'rowGap'	Taille de l'espace entre le texte et les traits de lignes du tableau
'colGap'	Taille de l'espace entre le texte et les traits de colonnes du tableau
'xPos' (1)	Alignement du repère virtuel du tableau dans la page : 'left', 'right', 'center', 'centre'. Par défaut 'center'. Ou aussi abscisse.
'xOrientation' (1)	Alignement du tableau par rapport à ce repère : 'left', 'right', 'center', 'centre'. Par défaut 'center'.
'width'	Taille exacte du tableau. Les cellules du tableau seront ajustées en conséquence.
'maxWidth'	Taille maximum du tableau. Les cellules du tableau seront ajustées en conséquence seulement si le tableau devait dépasser la taille maximum spécifiée.
'cols'	Tableau associatif dont les clés sont les nom de colonne (avant renommage) et les valeurs des tableaux associatifs de clés : 'justification' (de valeur 'left', 'right', 'center' : alignement du texte dans la colonne) et 'width' : taille de la colonne.

(1) : les options 'xPos' et 'xOrientation' s'utilisent ensemble.

Tableaux (V)

```
<?php
include 'class.ezpdf.php';

$pdf =& new Cezpdf();
$pdf->Cezpdf('a4','portrait');
$pdf->selectFont('./fonts/Helvetica.afm');

$data = array(
    array(1,'gandalf','wizard'),
    array(2,'bilbo','hobbit'),
    array(3,'frodo','hobbit'),
    array(4,'saruman','bad dude'),
    array(5,'sauron','really bad dude')
);

$title = "Lord of the ring characters";
```



1	gandalf	wizard
2	bilbo	hobbit
3	frodo	hobbit
4	saruman	bad dude
5	sauron	really bad dude

```
$options = array(
    'showLines'=> 2,
    'showHeadings' => 0,
    'shaded'=> 2,
    'shadeCol' => array(0.1,0.8,0.5),
    'shadeCol2' => array(1,0.4,0.4),
    'fontSize' => 12,
    'textCol' => array(0,0,0),
    'titleFontSize' => 16,
    'titleGap' => 8,
    'rowGap' => 5,
    'colGap' => 10,
    'lineCol' => array(1,1,1),
    'xPos' => 'left',
    'xOrientation' => 'right',
    'width' => 300,
    'maxWidth' => 300
);

$pdf->ezTable($data,$cols,$title,$options);

$pdf->ezStream();
?>
```

Dessin (I)

setColor(int r, int g, int b [, int force=0])

Change la valeur de la couleur texte courante. Prend en paramètre des trois composantes RVB (rouge, vert, bleu) de la couleur désirée. Les valeurs doivent être comprises entre 0 et 1.

setStrokeColor(int r, int g, int b [, int force=0])

Change la valeur de la couleur de trait courante. Idem que précédemment.

setLineStyle([int width] [, string cap] [, string join] [, array of int dash] [, int phase])

Change le style de trait. Largeur du trait : **width**. Type d'extrémité du trait : **cap** (valeurs possibles : 'butt', 'round', 'square'). **join** (valeurs possibles : 'miter', 'round', 'bevel'). Motif du trait : **dash**, c'est un tableau d'entiers alternant la taille du trait plain avec rien. Point de début du motif : **phase**.

line(int x1, int y1, int x2, int y2)

Dessine une ligne entre les points de coordonnées respectives (**x1,y1**) et (**x2,y2**) en utilisant les styles définis par **setLineStyle**.

Dessin (II)

curve(int x0, int y0, int x1, int y1, int x2, int y2, int x3, int y3)

Trace une courbe de Bezier. Les extrémités sont les points (x0,y0) et (x3,y3) et les points de contrôle sont les deux restants.

ellipse(int x0, int y0, int r1 [, int r2=0] [, int angle=0] [, int nSeg=8])

Trace une ellipse de centre (x0,y0) et de rayons r1 et r2 formée de nSeg courbes de Bezier. Le sens de rotation est angle (en degré dans le sens trigonométrique).

polygon(int p, int np [, int f=0])

Trace un polygone de np points dont le tableau p contient les coordonnées. Si f vaut 1 alors, la surface d'interne au polygone est remplie.

filledRectangle(int x, int y, int width, int height)

Trace un rectangle de coin inférieur gauche (x, y), de largeur width et de hauteur height. La couleur de remplissage est définie par **setStrokeColor()**.

Dessin (III)

rectangle(int x, int y, int width, int height)

Trace un rectangle de coin inférieur gauche (**x**, **y**), de largeur **width** et de hauteur **height**.

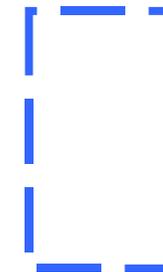
Exemple 1 :

```
$pdf->setStrokeColor(1, 0, 0);           // couleur rouge  
$pdf->setLineStyle(3, 'round', "", array(20, 10, 50)); // style  
$pdf->line(100, 200, 300, 600);        // line
```



Exemple 2 :

```
$pdf->setStrokeColor(0, 0, 1);           // couleur bleu  
$pdf->setLineStyle(1, "", "", array(10, 5)); // style  
$pdf->rectangle(100, 100, 30, 60); // rectangle
```



saveState() et **restoreState()** sauvegarde et restaurent respectivement l'état graphique (couleurs de remplissage et de fond ; style de trait).

Dessin (IV)

Quelques exemples de styles fournis par le manuel **readme.pdf** :

```
$pdf->setLineStyle(1);
```



```
$pdf->setLineStyle(5);
```



```
$pdf->setLineStyle(5, 'round');
```



```
$pdf->setLineStyle(5, '', '', array(5));
```



```
$pdf->setLineStyle(5, '', '', array(10, 5));
```



```
$pdf->setLineStyle(5, '', '', array(20, 5, 10, 5));
```



```
$pdf->setLineStyle(5, 'round', '', array(0, 15));
```



Images

addImage(int img, int x, int y, int w=0 [, int h=0] [, int quality=75])

Insert une image définie par le handler **img** de la bibliothèque graphique GD à la position (**x**, **y**). Les valeurs **w** et **h** définissent respectivement la largeur et la hauteur de l'image. Si une seule des deux est spécifiée, alors l'autre est calculée automatique afin de conserver les proportions d'origine. Les unités utilisées ne sont pas le pixel mais l'unité courante du document PDF.

Attention : cette méthode là nécessite la présence de la bibliothèque GD.

addJpegFromFile(string imgFileName, int x, int y, int w=0 [, int h=0])

Idem que précédemment mais **imgFileName** est le nom du fichier image au format Jpeg à insérer.

addPngFromFile(string imgFileName, int x, int y, int w=0 [, int h=0])

Idem mais avec le format PNG.

Exemple :

```
$pdf->addFromFile("images/mylogo.jpg", 100, 600);
```

Numérotation des pages (I)

ezStartPageNumbers(int x, int y, int size [, string pos] [, string pattern] [, int num])

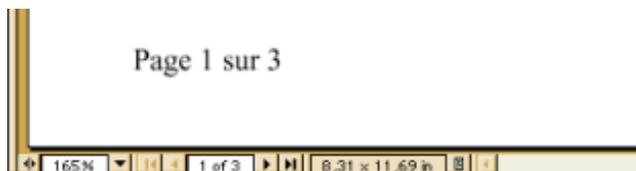
Affiche le numéro de page à la position (**x**, **y**) en l'alignant. La taille du texte est de **size** et est aligné à **pos** ('left' ou 'right') de la position. La numérotation peut être commencée arbitrairement à partir de **num**. Le texte affiché est donné par **pattern** qui peut contenir les mots clés suivants : {PAGENUM} et {TOTALPAGENUM}. La numérotation s'applique à partir de la page en cours.

ezStopPageNumbers([int stopTotal=0], [int next], [int setNum=0])

Permet d'interrompre la numérotation des pages.

Exemple :

```
$pdf->ezStartPageNumbers(100, 30, 12, 'left', 'Page {PAGENUM} sur {TOTALPAGENUM}');
```



Numérotation des pages (II)

int num = ezWhatPageNumber(int pageNum, [int setNum])

Retourne le numéro de la page courante selon la numérotation choisie.

Objets (I)

int id = openObject()

Commence un objet indépendant du document PDF identifié par l'identifiant **id**.
Le reste du code, jusqu'à **closeObject()** fera parti de cet objet.

reopenObject(int id)

Permet de rajouter des éléments à un objet (ou une page) déjà fermé mais dont on connaît l'identifiant **id**.

closeObject()

Ferme l'objet en cours.

Objets (II)

addObject(int id [, options='add'])

Ajoute l'objet identifié par **id**. On peut spécifier en **options** les paramètres décrits dans le tableau ci-après :

Valeur	Description
'add'	Ajout à la page en cours seulement
'all'	Ajout à toutes les pages depuis celle en cours
'odd'	Ajout à toutes les pages paires depuis celle en cours
'even'	Ajout à toutes les pages impaires depuis celle en cours
'next'	Ajout seulement à la page suivante
'nextodd'	Ajout à toutes les pages paires depuis la suivante
'nexteven'	Ajout à toutes les pages impaires depuis la suivante

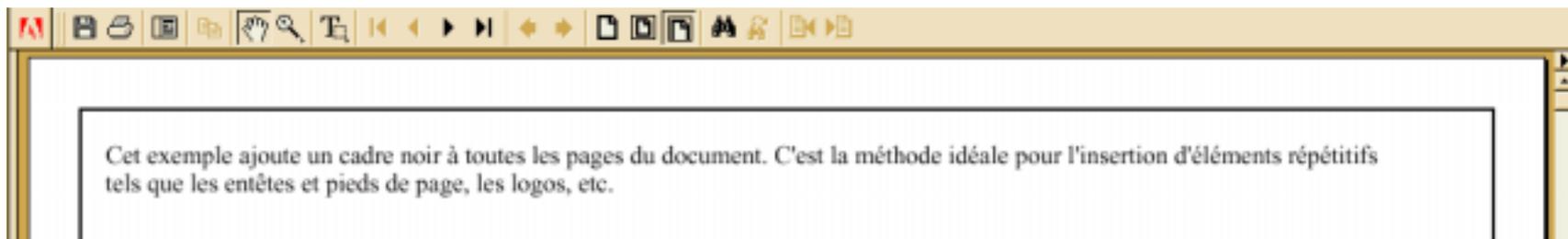
stopObject(id)

Si un objet identifié par **id** devait apparaître dans la page en cours, alors il n'apparaîtrait plus dans les suivantes.

Objets (III)

Exemple :

```
$id = $pdf->openObject();           // création de l'objet
$pdf->saveState();                 // sauvegarde de l'état graphique
$width = $pdf->ez['pageWidth'];     // largeur de page
$height = $pdf->ez['pageHeight'];  // hauteur de page
$pdf->setStrokeColor(0,0,0);       // couleur de trait : noir
$pdf->setLineStyle(1 , 'round', 'round'); // définition du style de trait
$pdf->rectangle(20,20,$width-40,$height-40); // création d'un rectangle
$pdf->restoreState();              // restauration de l'état graphique
$pdf->closeObject();               // fermeture de l'objet
$pdf->addObject($id, 'all');       // ajout de l'objet à toutes les pages
```



Code du document PDF

string s = output([int debug=0])

Retourne dans une chaîne de caractères le code du document PDF sans envoyer d'entête au navigateur. L'option **debug** mise à 1 empêche la compression.

Le code retourné ressemble à cela :

```
%PDF-1.3 %âãÏÓ 1 0 obj << /Type /Catalog /Outlines 2 0 R /Pages 3 0 R >> endobj 2 0 obj
<< /Type /Outlines /Count 0 >> endobj 3 0 obj << /Type /Pages /Kids [6 0 R 9 0 R 11 0 R
] /Count 3 /Resources << /ProcSet 4 0 R /Font << /F1 8 0 R >> >> /MediaBox [0 0 598
842] >> endobj 4 0 obj [/PDF /Text ] endobj 5 0 obj << /Creator (R and OS php pdf
writer, http://www.ros.co.nz) /CreationDate (D:20020508) >> endobj 6 0 obj << /Type
/Page /Parent 3 0 R /Contents 7 0 R >> endobj 7 0 obj << /Length 1440 >> stream 0.000
0.000 rg BT 30.000 789.680 Td /F1 20.0 Tf ( 0000002140 00000 n 0000003502 000000003567
00000 n trailer << /Size 13 /Root 1 0 R /Info 5 0 R >> startxref 3640 %% EOF
```

Il peut alors être sauvegardé sur le serveur dans un fichier .pdf pour réutilisation ultérieure sans avoir à le régénérer.

string s = ezOutput([int debug=0])

Très similaire à **output()** mais termine les tâche en cours avant de retourner le code (la numérotation des pages par exemple). De plus, si vous utilisez des fonctions de la classe EZPDF, utilisez **ezOutput()** plutôt que **output()** de la classe de base PDF.

Envoie du document au navigateur

stream([array options])

Provoque la compilation du document PDF et l'envoi au navigateur avec les entêtes adéquates.

Le tableau associatif **options** peut contenir les champs suivants :

Clé	Valeurs
'Content-Disposition'	Chaîne de caractères : nom du fichier à renvoyer au navigateur. utile aux anciens navigateur de reconnaître.
'Accept-Ranges'	Entier : 0 ou 1. 1 force l'envoi des entêtes HTTP au navigateur
'compress'	Entier : 0 ou 1. 1 force la compression (par défaut).

ezStream([array options])

Idem que **stream()** mais termine les tâche en cours avant de retourner le code (la numérotation des pages par exemple). De plus, si vous utilisez des fonctions de la classe EZPDF, utilisez **ezStream()** plutôt que **stream()** de la classe de base PDF.

Informations sur le document

addInfo(string|array string label, string value)

Permet de spécifier la valeur de certaines variables du document. Les valeurs de **label** peuvent être : 'Title', 'Author', 'Subject', 'Keywords', 'Creator', 'Producer', 'CreationDate', 'ModDate', 'Trapped'.

Depuis la version 003, **label** peut être un tableau associatif. Dans ce cas **value** n'est pas utilisé.

Exemple 1 :

```
$pdf->addInfo('Author', 'Hugo Etiévant');
```

Exemple 2 :

```
$infos = array('Title' => 'EZPDF Tutorial', 'CreationDate' => '2002/04/26' );  
$pdf->addInfo($infos);
```

Préférences pour le plugin (I)

addPreferences(string|array string label, string value)

Permet de spécifier la valeur de certaines variables du plugin de visualisation. Les valeurs de **label** peuvent être :

'HideToolbar',

'HideMenubar',

'HideWindowUI',

'FitWindow',

'CenterWindow',

'NonFullScreenPageMode',

'Direction'.

Depuis la version 003, **label** peut être un tableau associatif. Dans ce cas **value** n'est pas utilisé.

Préférences pour le plugin (II)

`openHere(string style [, int a] [, int b] [, int c])`

Permet de forcer l'ouverture du document à une page en particulier et de forcer le mode de visualisation. Cette méthode doit être appelée dans la page en question.

Exemples :

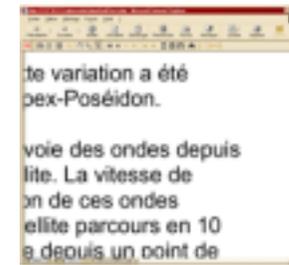
`$pdf->openHere('XYZ', 600, 800, 5);`

`$pdf->openHere('Fit');`

Aperçu des modes d'ouverture possibles :



Fit



XYZ



FitH



FitV



FitB



FitBH



FitBV

Protection par mot de passe

`setEncryption([userPass=""] [, ownerPass=""] [, pc=array])`

Cette méthode permet de protéger le document par mot de passe. Il faut lui spécifier le mot de passe de l'utilisateur **userPass**, celui du propriétaire **ownerPass** ainsi que les actions autorisées pour l'utilisateur. Ces actions peuvent être les copier/coller de texte (**'copy'**) et l'impression (**'print'**).

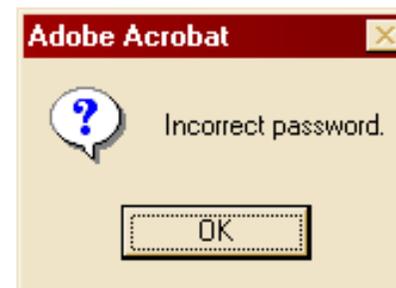
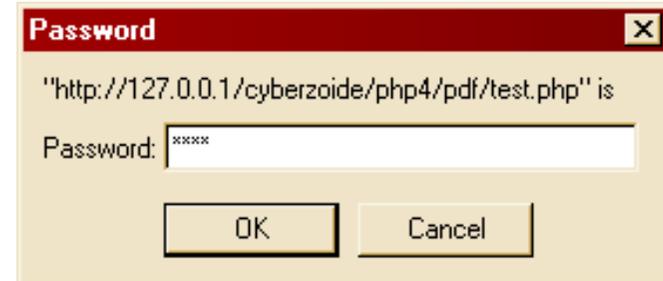
Le propriétaire a tout les droits.

L'utilisateur n'aura que les droits alloués explicitement.

Exemple :

`$pdf->setEncryption('toto', 'titi', array('copy'));`

Dans cet exemple, si on entre le mot de passe **'toto'** dans la boite de dialogue affichée par Acrobat Reader on pourra lire le document, sinon l'accès sera refusé.



Transactions

transaction(action)

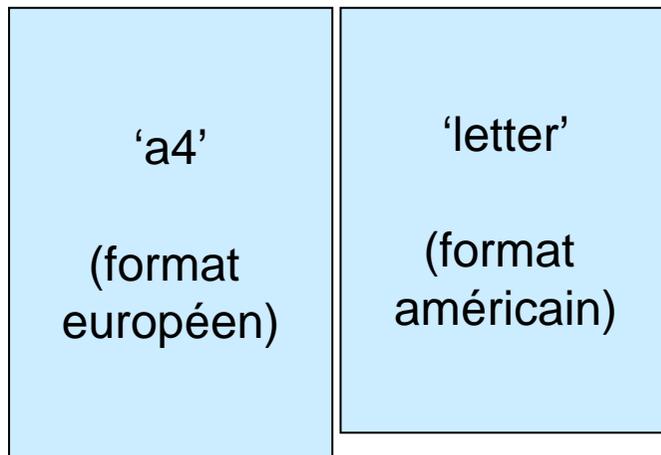
Cette méthode permet d'annuler des opérations sur votre document comme cela peut se faire sur les bases de données. Les actions possibles sont listées dans le tableau ci-après :

Action	Description
'start'	Marque l'état vers lequel on va retourné en cas de 'abord'.
'commit'	Valide toutes les modifications effectuée depuis le 'start'. Marque l'état actuel comme dernier état vers lequel retourner en cas de 'abord' (équivalent au COMMIT de Oracle).
'rewind'	Equivalent à un 'abord' puis un 'start' mais plus performant.
'abord'	Supprime toutes les modifications apportées depuis le dernier état (équivalent au ROLLBACK de Oracle).

Unité utilisée

L'unité utilisée par défaut est le point et équivaut à 1/72 de inch (vaut aussi 0.3528 millimètre).

Le format A4 (210 x 297 mm) fait donc 595.28 x 841.89 points.



Historique

- ▶ **12 février 2003** : ajout des nouvelles fonctionnalités de la version 009c (47 diapos)
- ▶ **13 mai 2002** : mise en ligne de la version 007 (34 diapos)
- ▶ **26 avril 2002** : création du document par Hugo Etiévant

Merci de faire parvenir vos suggestions et critiques à l'auteur afin de contribuer à l'amélioration de la qualité de ce document.

Hugo Etiévant
cyberzoide@yahoo.fr
<http://cyberzoide.developpez.com/>