

# Définition de la classe DataPdf

---

## Nom de la classe

Tests\Rcnchris\Core\PDF\Behaviors\DataPdf

## Parent

Tests\Rcnchris\Core\PDF\DocPdf

## 0 classes implémentées

## 1 traits utilisés

Rcnchris\Core\PDF\Behaviors\DataPdfTrait

## 148 méthodes publiques

Header, Footer, demo, printPublicProperties, printPublicMethods, printNativeMethods, \_\_construct, AddPage, \_\_toString, getTotalPages, getBodySize, getMargin, setMargin, getCursor, setCursor, getOrientation, getFonts, hasFont, getToolColor, getMetadata, setMetadata, toFile, SetFont, getFontProperty, getLinks, fileToPdf, hexaToRgb, setToolColor, setWriter, SetMargins, SetLeftMargin, SetTopMargin, SetRightMargin, SetAutoPageBreak, SetDisplayMode, SetCompression, SetTitle, SetAuthor, SetSubject, SetKeywords, SetCreator, AliasNbPages, Error, Close, PageNo, SetDrawColor, SetFillColor, SetTextColor, GetStringWidth, SetLineWidth, Line, Rect, AddFont, SetFontSize, AddLink, SetLink, Link, Text, AcceptPageBreak, Cell, MultiCell, Write, Ln, Image, GetPageWidth, GetPageHeight, GetX, SetX, GetY, SetY, SetXY, Output, \_dochecks, \_checkoutput, \_getpagesize, \_beginpage, \_endpage, \_loadfont, \_isascii, \_httpencode, \_UTF8toUTF16, \_escape, \_textstring, \_dounderline, \_parsejpg, \_parsepng, \_parsepngstream, \_readstream, \_readint, \_parsegif, \_out, \_put, \_getoffset, \_newobj, \_putstream, \_putstreamobject, \_putpage, \_putpages, \_putfonts, \_tounicodecmap, \_putimages, \_putimage, \_putxobjectdict, \_putresourcedict, \_putresources, \_putinfo, \_putcatalog, \_putheader, \_puttrailer, \_enddoc, getBookmarks, addBookmark, getBookmarksMaxLevel, putBookmarks, setJoinedPane, joinFile, putJoinedFiles, createIndex, alert, title, codeBloc, printLink, printInfoClass, printDocumentProperties, getTitleTemplates, setTitleTemplates, puce, draw, addLine, roundedRect, arc, circle, ellipse, grid, ean13, upca, code39, barcode, getBarcodeCheckDigit, testBarcodeCheckDigit, hasForm, tree, toView, toDownload, setData, getData, hasKey, hasValue

---

# Propriétés publiques propres à DocPdf

---

## options

---

Collection qui contient les propriétés d'écritures.

### *Exemple*

```
$pdf->options->get('fontFamily')
```

### *Retourne*

```
helvetica
```

### *Liste des clés*

```
orientation, unit, format, rotation, marges, fontFamily, fontStyle, fontSize, underline,  
heightline, border, align, fill, ln, textColor, fillColor, drawColor
```

# Méthodes publiques

---

## **\_\_construct**

Il s'agit du constructeur de la classe. Il permet de fixer le format des pages, leur orientation par défaut ainsi que l'unité de mesure utilisée dans toutes les méthodes (sauf pour les tailles de police).

[Voir la documentation sur le site](#)

## *Paramètre*

```
array|null $options Options par défaut de construction du document
```

## *Retourne*

```
Rcnchris\Core\PDF\AbstractPDF
```

## *Surcharge*

Utilisation de préférences par défaut pour créer le document (police, hauteur de ligne...). Elles sont stockées dans la propriété 'options' qui est une Collection.

## *Exemple*

```
$pdf = new DocPdf(['orientation' => 'L']);
```

## **\_\_toString**

---

Obtenir le document au format string.

### *Retourne*

```
string
```

### *Exemple*

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

## AddPage

Ajoute une nouvelle page au document. Si une page était en cours, la méthode Footer() est appelée pour traiter le pied de page. Puis la page est ajoutée, la position courante mise en haut à gauche en fonction des marges gauche et haute, et Header() est appelée pour afficher l'en-tête.

La police qui était en cours au moment de l'appel est automatiquement restaurée. Il n'est donc pas nécessaire d'appeler à nouveau SetFont() si vous souhaitez continuer avec la même police. Même chose pour les couleurs et l'épaisseur du trait. L'origine du système de coordonnées est en haut à gauche et les ordonnées croissantes vont vers le bas.

[Voir la documentation sur le site](#)

## Surcharge

Application des préférences par défaut, crée l'alias du nombre de page à la première page ajoutée.

## getTotalPages

---

Obtenir le nombre total de pages au moment de l'appel.

### *Retourne*

```
int
```

### *Exemple*

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

### *Résultat*

```
i:6;
```

## getBodySize

---

Obtenir la taille du corps.

### *Paramètre*

```
string|null $type (width ou height)
```

### *Retourne*

```
array|double
```

### *Exemple*

```
$pdf->getBodySize('width');
```

### *Résultat*

```
d:189.99905555555551;
```

## getOrientation

---

Obtenir l'orientation courante.

### *Retourne*

```
string
```

### *Exemple*

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

### *Résultat*

```
s:1:"P";
```



## getMargin

---

Obtenir toutes les marges ou l'une d'entre elle.

### *Paramètre*

```
string|null $type (r, l, t, b)
```

### *Retourne*

```
array|double
```

### *Exemple*

```
$pdf->getMargin('b');
```

### *Résultat*

```
d:20.002499999999998;
```

## setMargin

---

Définir la valeur d'une marge.

### *Paramètres*

```
string $type  Type de marge (top, bottom, left, right)
double $value Valeur de la marge
```

### *Retourne*

```
Rcnchris\Core\PDF\AbstractPDF
```

### *Exemple*

```
$pdf->setMargin('left', 15);
```

## getCursor

---

Obtenir toutes les marges ou l'une d'entre elle.

### *Paramètre*

```
string|null $type (r, l, t, b)
```

### *Retourne*

```
array|double|boolean
```

### *Obtenir la position de x et y*

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

### *Résultat*

```
a:2:{s:1:"x";d:10.001249999999999;s:1:"y";d:118.00125;}
```

## getMetadata

---

Obtenir les meta données ou la valeur de l'une d'entre elle.

### *Paramètre*

```
string|null $name Nom de la clé à retourner
```

### *Retourne*

```
array|bool
```

### *Obtenir toutes les meta-données*

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

### *Résultat*

```
a:4:{s:7:"Creator";s:7:"My Core";s:6:"Author";s:3:"rcn";s:5:"Title";s:43:"Tests unitaires du 29-04-2018 à 08:46:04";s:7:"Subject";s:57:"Tests unitaires Tests\Rcnchris\Core\PDF\Behaviors\DataPdf";}
```

### *Obtenir l'auteur du document*

```
$pdf->getMetadata('Author');
```

### *Résultat*

```
s:3:"rcn";
```

## setMetadata

---

Définir une meta-donnée.

### *Paramètre*

```
string      $key    Nom de la clé ou tableau  
mixed|null $value  Valeur de la clé
```

### *Retourne*

```
void
```

### *Définir une meta-donnée*

```
$pdf->setMetadata('Test', 'Ola le test');
```

### *Définir plusieurs meta-avec un tableau*

```
$pdf->>setMetadata([  
    'Ola' => 'le test',  
    'Ole' => 'On ferme'  
]);
```

## setCursor

---

Se positionner dans le document.

### *Paramètre*

```
int      $x  
int|null $y
```

### *Retourne*

```
Rcnchris\Core\PDF\AbstractPDF
```

*Se positionner à la colonne 20 et à la ligne 55.*

```
$pdf->setCursor(20, 55);
```

## hasFont

---

Vérifier la présence d'une police par son nom.

### *Retourne*

```
boolean
```

### *La police 'helvetica' est-elle disponible ?*

```
$pdf->hasFont('helvetica');
```

### *Résultat*

```
b:1;
```

## getFonts

---

Obtenir la liste des polices disponibles.

### *Retourne*

```
array
```

### *Obtenir la liste des polices*

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

### *Résultat*

```
a:5:{i:0;s:7:"courier";i:1;s:9:"helvetica";i:2;s:5:"times";i:3;s:6:"symbol";i:4;s:12:"zapfdingbats";}
```



## getToolColor

---

Obtenir la commande de la couleur courante d'un outil.

### *Paramètre*

```
string|null $tool text, fill ou draw
```

### *Retourne*

```
array|string|bool
```

### *Obtenir toutes les commandes*

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

### *Résultat*

```
a:3:{s:4:"draw";s:20:"0.800 0.800 0.800 RG";s:4:"fill";s:20:"0.800 0.800 0.800  
rg";s:4:"text";s:7:"0.000 g";}
```

### *Obtenir la commande de l'outil de remplissage*

```
$pdf->getToolColor('fill');
```

### *Résultat*

```
s:20:"0.800 0.800 0.800 rg";
```

## toFile

---

Enregistrer le document PDF sur le serveur.

### *Paramètre*

```
string|null $fileName Chemin et nom du fichier PDF (sans l'extension)
```

### *Retourne*

```
string
```

### *Exemple*

```
$pdf->toFile('path/to/file/filename');
```

# Méthodes natives à FPDF non surchargées

## AcceptPageBreak

Lorsqu'une condition de saut de page est remplie, la méthode est appelée, et en fonction de la valeur de retour, le saut est effectué ou non.

L'implémentation par défaut renvoie une valeur selon le mode sélectionné par SetAutoPageBreak(). Cette méthode est appelée automatiquement et ne devrait donc pas être appelée directement par l'application.

[Voir la documentation sur le site](#)

### *Retourne*

```
boolean
```

### *Exemple*

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

### *Résultat*

```
b:1;
```

## AddFont

Importe une police TrueType, OpenType ou Type1 et la rend disponible.

Il faut au préalable avoir généré un fichier de définition de police avec l'utilitaire MakeFont. Le fichier de définition (ainsi que le fichier de police en cas d'incorporation) doit être présent dans le répertoire des polices. S'il n'est pas trouvé, l'erreur "Could not include font definition file" est renvoyée.

[Voir la documentation sur le site](#)

### *Retourne*

```
void
```

### *Exemple*

```
$pdf->AddFont( 'Comic', 'I' )
```

## SetMargins

Fixe les marges gauche, haute et droite. Par défaut, elles valent 1 cm. Appelez cette méthode si vous désirez les changer.

[Voir la documentation sur le site](#)

*Retourne*

```
void
```

*Exemple*

```
$pdf->SetMargins(15, 20)
```

## SetLeftMargin

---

Fixe la marge gauche. La méthode peut être appelée avant de créer la première page.

Si l'abscisse courante se retrouve hors page, elle est ramenée à la marge.

[Voir la documentation sur le site](#)

*Retourne*

```
void
```

*Exemple*

```
$pdf->SetLeftMargin(15)
```