









# Manuel de Prawn-for-book

*(le manuel autoproduit de l'application  
de mise en forme professionnelle)*

Philippe Perret

AVANT-PROPOS .....	9
FONCTIONNALITÉS .....	13
Généralités .....	15
Aperçu rapide des commandes .....	15
Les Forces de <i>Prawn-For-Book</i> .....	15
Définitions minimales .....	17
Définition des marges .....	18
Les Recettes .....	21
Recette du livre .....	21
Recette de la collection .....	21
Définition de la fonte par défaut .....	22
Éditeur / Maison d'édition .....	22
Aides & Assistants .....	23
Manuel et autres aides .....	23
Les « Snippets » .....	24
Messages d'erreurs et notices .....	25
Exportation du texte .....	25
Format du livre et des pages .....	27
Page en multi-colonnes .....	27
Le Mode Expert .....	30
Description .....	30
Indication du contexte d'erreur .....	30
Injection .....	30
Évaluation du code ruby .....	31
Bibliographies en mode expert .....	32
Mode Multi-colonnes .....	32
ANNEXE .....	35
Reconstruction du manuel .....	36
Marques markdown .....	36
Le format YAML de la recette .....	36
Les "Fontes-Strings" .....	37
Définition de la couleur .....	37
Rogner une image SVG .....	39
Synopsis de création .....	42

Installation de l'application .....	42
Constantes .....	42
Liste exhaustive des fonctionnalités .....	43





# Avant-propos

Ce manuel présente toutes les fonctionnalités, à jour, de l'application « Prawn-for-book » (« Prawn pour les livres »), application dont la principale vocation est d'**obtenir un document PDF professionnel prêt pour l'imprimerie** à partir d'un simple fichier de texte (contenant le contenu du livre, le roman par exemple).

Ce manuel de 50 pages est auto-produit par « **Prawn-for-book** », c'est-à-dire qu'il est construit de façon *programmative* par l'application elle-même. Il en utilise toutes les fonctionnalités puisqu'il génère de façon automatisé les exemples et notamment les *helpers* de mise en forme, les références croisées ou les bibliographies. En ce sens, ce manuel sert donc aussi de test complet de l'application puisqu'une fonctionnalité qui ne fonctionnerait pas ici ne fonctionnerait pas non plus dans le livre produit.

Si vous êtes intéressé(e) de voir comment il est généré, vous pouvez consulter principalement le fichier markdown `texte.pfb.md`, le fichier ruby `prawn4book.rb` et le fichier recette yaml `recipe.yaml` qui le définissent, dans le dossier `Manuel/manuel_building` de l'application.





# Fonctionnalités



## Aperçu rapide des commandes

Si vous avez déjà consulté l'aide et produit votre premier livre, vous aurez peut-être simplement besoin d'un rafraichissement de mémoire sur les commandes de base. Ce chapitre les expose.

*(note : toutes ces commandes s'effectuent dans une console/un terminal ouvert dans le dossier du livre à considérer)*

<b>pfb init</b>	Instanciation d'un nouveau livre/une nouvelle collection.
<b>pfb build</b>	Construction du livre.
<b>pfb build -open</b>	Construction et ouverture du livre.
<b>pfb build -t</b>	Exportation du livre
<b>pfb open</b>	Ouvrir un élément quelconque.
<b>pfb install</b>	Installer le livre (les snippets).

## Options de la commande **build**

La commande de construction du livre (**build**) supporte plusieurs options qui affectent son comportement :

<b>-open</b>	Ouvrir le livre une fois construit.
<b>-display_margins</b>	Construit le livre avec les marges visibles.
<b>-grid</b>	Construit le livre avec les lignes de référence.
<b>-debug</b>	Affiche des messages d'erreur plus précis.

## Les Forces de *Prawn-For-Book*

**Prawn-for-book** possède de nombreuses forces et de nombreux atouts dont nous ne pouvons que donner un aperçu non exhaustif dans cette introduction.

- Sans configuration ou définitions, l'application produit un document PDF valide pour l'imprimerie professionnelle respectant toutes les normes et les habitudes en vigueur, avec par exemple toutes les lignes de texte alignées sur une grille de référence, avec une mise en page ne contenant aucune veuve, aucune orpheline et aucune ligne de voleur,
- permet d'utiliser de façon simple les feuilles de styles,
- respecte toutes les règles de typographie en vigueur,
- produit une page de titre valide par défaut,
- permet de travailler sur une collection entière,
- permet de gérer automatiquement une infinité d'index,
- gère facilement les bibliographies,
- gère les références, même à d'autres livres,
- gère les images flottantes,

- produit bien sûr une table des matières valide par défaut,
- extension infinie pour les experts qui connaissent le langage Ruby et la gem Prawn.

Bien que **Prawn-For-Book** soit capable de produire de façon automatique ce document, l'application offre la possibilité de définir tous les éléments de façon très précise et très fine pour obtenir le rendu exact souhaité, grâce à une *recette* qui accompagne le texte et où peuvent être paramétrés tous les aspects du livre.

Plus loin encore, **Prawn-For-Book** permet de travailler comme aucun autre logiciel sur **une collection entière**, grâce à un *fichier recette* partagé par tous les livres — chacun pouvant définir sa propre recette pour rectifier des aspects particuliers. De cette manière, on peut très simplement obtenir une collection cohérente partageant la même charte. On peut même obtenir des références croisées dynamiques entre les différents livres.

Le reste de ce manuel vous permettra de découvrir l'ensemble des fonctionnalités à votre disposition.



# Définitions minimales

Cette section se concentre sur les définitions minimales que vous aurez à faire si les définitions par défaut ne conviennent pas au livre que vous êtes en train d'écrire. Typiquement, il est fort possible que la taille du livre ainsi que les marges ne soient pas définies par défaut comme votre livre le demande.

C'est ici que vous apprendrez à régler ces valeurs.

Cette partie suppose que vous avez déjà parcouru cette aide et que vous connaissez, notamment, l'usage des recettes utilisées par **Prawn-For-Book** (voir « recette du livre », page 21 et « recette de la collection », page 21). Dans le cas contraire, reportez-vous à ces sections avant de lire celle-ci (qui se trouve avant juste pour pouvoir l'atteindre plus rapidement lorsque l'on connaît déjà l'application et que l'on a juste besoin de se rafraichir la mémoire.

## Définition des titre et auteur

La première définition concernant le livre sera certainement son titre et son auteur(e) que nous ne pouvons pas définir par défaut. Pour ce faire, nous définissons les propriétés **title** (« titre » en anglais) et **author** (« auteur » en anglais) dans la section **book\_data** (« données du livre » en anglais) de la recette du livre `recipe.yaml`.

```
# ./recipe.yaml
book_data:
  title: "Le titre du livre"
  author: "Prénom NOM"
```

Notez que les noms se mettent avec capitales, pour que **Prawn-For-Book** puisse les reconnaître.

## Taille du livre

## Marges du livre

## Polices du livre

## Définition des marges

Des marges par défaut sont proposées, mais vous pouvez tout à fait définir celles que vous voulez très précisément, dans la recette du livre ou de la collection.

La seule chose à comprendre, par rapport aux documents dont vous avez l'habitude, c'est qu'ici les pages sont doubles, en vis-à-vis, et définissent donc :

- une marge haute et une marge basse (traditionnelle),
- une marge intérieure, qui comme son nom l'indique est à l'intérieur du livre, contre le dos,
- une marge extérieure, qui comme son nom l'indique est tournée vers l'extérieur du livre, vers la tranche (attention, la tranche du livre, ça n'est pas la partie qui reprend le titre du livre et son auteur, qu'on appelle le « dos »).

Ces marges sont donc définies par les propriétés `top` (haut) `bot` (pour « bottom », bas), `ext` (pour « extérieur ») et `int` (pour « intérieur »).

Traditionnellement, la marge intérieure est plus large que la marge extérieure, car une bonne partie de cette marge est prise dans la reliure.

De la même manière, la marge basse est plus large que la marge haute car elle contient le numéro de page. Il peut cependant arriver que la marge haute contienne un entête.

Pour régler parfaitement les marges, vous pouvez soit ajouter l'option `-display_margins` à la commande `pfb build` qui construit le livre, soit mettre le `show_margins` de la recette à `true`, comme nous l'avons fait ci-dessous.

Dans l'exemple ci-dessous nous avons volontaire *pousser* les valeurs pour rendre bien visibles les changements.

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```
book_format:
  page:
    margins:
      top: 60mm
      bot: 70mm
      ext: 2mm
      int: 30mm
    show_margins: true
```

*Le livre final (document PDF) contiendra :*

---

Pour cette page, où les marges sont visibles, on illustre des marges de page à 60mm en haut, 70mm en bas, 30mm à l'intérieur et 2mm à l'extérieur.

Vous remarquez donc une immense marge en bas, une grande marge en haut, une marge externe toute petite (*ce qui ne serait pas du tout bon pour une impression de livre*) et une marge intérieure moyenne.

Remarquez aussi que le texte est automatiquement justifié, il s'aligne parfaitement sur la marge gauche et la marge droite, ce qui donne un rendu impeccable.

---



# Les Recettes

## Recette du livre

La *recette du livre* du livre est un fichier de nom `recipe.yaml` qui se trouve à la racine du dossier du livre. Son nom vient de « recipe » qui signifie *recette* en anglais et de `.yaml`, extension des fichiers au format simple YAML (cf. Le format YAML de la recette (p. 36)).

Vous pouvez voir ci-dessous un extrait du fichier recette de ce manuel (qui a bien sûr été produit par **Prawn-For-Book**).

*Extrait de fichier recette*

---

```
---
#<book_data>
book_data:
  title:      "Manuel de Prawn-for-book"
  author:     "Philippe PERRET"
  version:    1.3
  isbn:       null
  # ...
#</book_data>
#<book_format>
book_format:
  book:
    width: '210cm'
    height: '297cm'
    orientation: portrait
    format: :pdf
# ...
```

---

## Recette de la collection

La *recette de la collection* est un fichier de nom `recipe_collection.yaml` qui se trouve à la racine du dossier d'une collection de livres. Son nom vient de « recipe » qui signifie *recette* en anglais et de `.yaml`, extension des fichiers au format simple YAML (cf. page Le format YAML de la recette (p. 36)).

Vous pouvez trouver ci-dessous les données propres à une collection.

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```
---
# (données de la collection)
collection:
```

```

name: "Nom de la collection"
short_name: "Nom raccourci (pour les messages)"
#/collection
# ...

```

---

## Définition de la fonte par défaut

Bien que l'application propose, clé en main, une fonte qui peut être utilisée pour imprimer un livre, on peut définir n'importe quelle fonte comme fonte par défaut, et même une fonte personnelle dont on aura au préalable acheté la licence (c'est presque toujours nécessaire si le livre doit être vendu).

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```

book_format:
  text:
    default_font_and_style: "Times-Roman/"
    default_font_style: ""
    default_font_size: 20

```

*Le livre final (document PDF) contiendra :*

---

Pour écrire ce texte, nous avons ponctuellement modifié la police par défaut en utilisant la fonte Times-Roman avec une taille de 20 et le style .

---

## Éditeur / Maison d'édition

L'éditeur du livre (« publisher » en anglais) ou la maison d'édition se définissent dans la propriété `publisher` de la recette du livre ou de la collection.

On peut définir toutes les données utiles

{{TODO: Poursuivre cette fonctionnalités}}

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```

---
publisher:
  name: "<Éditeur / Maison d'édition>"
  adresse: "<Numéro rueCode Ville>"
  contact: "<contact@chez.lui>"
  url: "https://<url/site>"
  logo_path: "chemin/vers/logo"

```

---

# Aides & Assistants

## Manuel et autres aides

Tout au long de la conception de son livre — et sa production — on peut obtenir de l'aide sur **Prawn-For-Book** de plusieurs façon.

La façon la plus complète consiste à ouvrir ce *manuel autoproduit* qui contient l'intégralité des fonctionnalités de **Prawn-For-Book** expliquées de façon détaillée. C'est incontestablement **la bible de l'application**. Pour l'ouvrir, il suffit de jouer :

```
> pfb manuel
```

On peut obtenir une aide beaucoup plus succincte, rappelant les commandes de base, en jouant au choix l'une de ces commandes :

```
> pfb
> pfb aide
> pfb -h
> pfb --help
```

## Aide **Prawn-For-Book** en ligne de commande

On peut obtenir une aide rapide sur un sujet donné, ou un mot, ou une fonctionnalité, en développant la commande `pfb aide` :

```
> pfb aide "le mot ou l'expression recherché"
```

Après avoir lancé cette commande, **Prawn-For-Book** affiche tous les endroits du manuel qui contiennent l'expression recherchée (par pertinence) et permet de développer le passage.

## Rechercher régulière dans l'aide

On peut même faire une recherche *régulière* avec une *expression rationnelle* (si vous ne comprenez pas, cette fonctionnalités n'est peut-être pas pour vous...). L'expression rationnelle se trouvera entre guillemets et commencera et terminera avec une balance (« / »).

```
> pfb aide "/expression à rechercher>/"
```

Quelques exemples :

- Pour rechercher deux mots qui doivent se trouver dans la même phrase, et dans l'ordre donnée : `pfb aide "/mot1(.+ ?)mot2/"`.
- Pour chercher plusieurs mots : `pfb aide "/(mot1|mot2)/"`.
- Pour chercher un mot exact, mais qui peut être au pluriel : `pfb aide "/\bmots ?\b/"` (le `\b` désigne un délimiteur de mots).

## Les « Snippets »

Les « snippets »<sup>1</sup> (ou *complétions*) sont particulièrement utiles lorsque l'on désire gagner du temps dans la rédaction de son livre (même si, philosophiquement, "gagner du temps" est un concept *contre-artistique*...)

Ils consistent à taper quelques lettres, parfois une seule, de jouer la touche tabulation, et un texte vient remplacer automatiquement ces lettres.

Par exemple, si nous avons un personnage qui s'appelle *Rostatopoulos* et que nous ne voulons pas taper chaque fois ce prénom (avec toutes les erreurs de typo possibles...), nous créons un snippet avec la lettre « R » majuscule et chaque fois que nous tapons « R » suivi de la touche tabulation, **Prawn-For-Book** remplace ce « R » par « Rostatopoulos ».

---

<sup>1</sup> Les snippets ne sont pas à confondre avec les variables textuelles qui s'écrivent textuellement dans le texte du fichier et seront ensuite remplacées par leur valeur fixe ou dynamique. Un snippet est remplacé immédiatement dans le texte par sa valeur définie.

---

### Exclusivement dans Sublime Text

Mais cette fonctionnalité pratique ne fonctionne que dans un IDE adapté. En l'occurrence, à l'heure où nous écrivons ces lignes, elle ne fonctionne que dans l'application Sublime Text.

### Installer les snippets

Pour pouvoir fonctionner, les snippets doivent être *installés* dans l'éditeur, il ne suffit pas qu'ils soient définis. Chaque fois que vous travaillez sur un nouveau livre, il faut installer les snippets définis.

On installe ces *snippets* à l'aide de la commande (dans une console ouverte au dossier du livre) :

```
> pfb install
```

### Programmer un snippet

Programmer un snippet est simplissime, il suffit d'appeler la commande **pfb snippet**. Si l'on veut gagner du temps, il suffit de lui donner d'abord les lettres à taper (par exemple « R » dans notre exemple) suivi du texte de remplacement (« Rostatopoulos » dans notre exemple). Cela donnera :

```
> pfb snippet R Rostatopoulos
```

Si le texte de remplacement contient plusieurs mots, il est indispensable de les mettre entre guillemets.

```
> pfb snippet R "Rostatopoulos Alexis Triponov"
```



## Messages d'erreurs et notices

On peut signaler des erreurs (messages rouges) et des messages de notices (messages bleus) au cours de la construction du livre grâce, respectivement, aux méthodes `erreur(« message> »)` (ou `error(« message> »)`) et `notice(« message> »)`.

Ces méthodes doivent être placées seules sur une ligne, entre des doubles parenthèses.

`(( erreur(« message d'erreur> ») ))`

**Attention : ces messages ne seront jamais gravés dans le livre, ils n'apparaîtront qu'en console lorsque l'on construira le livre.**

### Position

Un des grands avantages de ces messages est qu'ils indiquent clairement la source de l'erreur ou de la note. Ils indiquent le numéro de page dans le livre, ainsi que le numéro de ligne dans le fichier source ou le fichier inclus. De cette manière, on retrouve très rapidement l'endroit concerné par la note ou l'erreur.

### Utilisation

On peut utiliser ces méthodes par exemple pour signaler une erreur dans le livre, qu'on ne peut pas corriger au moment où on la voit. Exemple :

`(( erreur(« Il manque ici le chapitre 13 ») ))`

Ou simplement pour signaler une chose qu'il faut garder en tête. Par exemple :

`(( notice(« Bien s'assurer que l'image qui suit soit en haut de la page. ») ))`

### Exemple

À cet endroit précis nous avons placé un appel à une note avec le code :

`(( notice(« Une note depuis le manuel. ») ))`

Vous devriez la voir si vous lancez la Reconstruction du manuel (p. 36).

#### *Exemple dans `texte.pfb.md`*

---

Un paragraphe.

`(( add_notice("Une simple notice pour l'exemple.") ))`

Un autre paragraphe.

`(( add_erreur("Une erreur signalée.") ))`

Un troisième paragraphe.

---

## Exportation du texte

Pour corriger orthographiquement le texte, par exemple avec Antidote, on peut bien sûr prendre le texte du fichier `texte.pfb.md` en faisant abstraction de tout ce qui relève de la mise en forme et des codes éventuels.

Mais lorsque ces codes « extra-texte » représentent une quantité non négligeable, il est préférable de procéder à un export du seul texte du livre.

Pour se faire, il suffit de jouer la construction du livre avec l'option `-t`.

```
> cd chemin/vers/mon/livre  
> pfb build -t
```

Le texte seul du livre est alors mis dans un fichier `only_text.txt` (« seulement le texte » en anglais) à la racine du livre. C'est ce fichier qu'il faut corriger.

# Format du livre et des pages

## Page en multi-colonnes

On peut provisoirement passer en double colonnes grâce à la marque :

```
(( colonnes(2) ))
```

Pour arrêter d'être en double colonnes, il suffit d'utiliser :

```
(( colonnes(1) ))
```

Vous l'aurez déjà compris, grâce à cette marque, on peut avoir autant de colonnes que l'on désire.

## Définition plus précise des colonnes

On peut définir la gouttière (espace entre chaque colonne) grâce à la propriété `gutter` à mettre en deuxième paramètre :

```
(( colonnes(2, gutter: 40) ))
```

On peut définir aussi sur quelle largeur les colonnes devront tenir, par exemple la moitié de la page :

```
(( colonnes(2, width: PAGE_WIDTH/2) )) 2
```

---

<sup>2</sup> Vous remarquez ci-dessus l'utilisation d'une constante (cf. Constantes (p. 42)).

---

## Précaution pour les colonnes

Attention à toujours terminer par `(( colonnes(1) ))`, surtout si c'est la dernière page, dans le cas contraire les pages multi-colonnes ne seraient pas gravées.

*Le livre final (document PDF) contiendra :*

---

Un premier paragraphe qui commence en simple colonne. Juste sous ce paragraphe, on a inscrit le code (invisible ici) : `(( colonnes(3) ))` qui permet de passer la suite en triple colonnes.

Début du texte. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut	officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In	mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est
--	---	--

dolor consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In mollit  
anim veniam est ut officia sit  
mollit est dolor consequat  
cillum. In mollit anim veniam  
est ut officia sit mollit est  
dolor consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In mollit

anim veniam est ut officia sit  
mollit est dolor consequat  
cillum. In mollit anim veniam  
est ut officia sit mollit est  
dolor consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In mollit  
anim veniam est ut officia sit  
mollit est dolor consequat  
cillum. In mollit anim veniam  
est ut officia sit mollit est

dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. In mollit anim veniam est ut officia sit mollit est dolor consequat cillum. Fin du texte.

On revient ensuite à un texte sur une colonne avec la marque (( colonnes(1) )). Et c'est la fin de l'usage des colonnes multiples, on revient sur une page normale.

La double colonne suivante est obtenue quant à elle grâce au code : `(( colonnes(2, width:PAGE_WIDTH/1.5, gutter:50) ))` qui est placé juste sous cette ligne.

Début du texte. In mollit  
anim veniam est ut officia  
sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In

mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. In  
mollit anim veniam est ut  
officia sit mollit est dolor  
consequat cillum. . Fin du  
texte.

On revient à nouveau à un texte sur une colonne avec la marque ( ( colonnes(1) ) ).

Par défaut, **Prawn-For-Book** laisse une ligne vide au-dessus et une ligne vide au-dessous d'un texte en multi-colonnes. On peut contrecarrer ce comportement à l'aide des propriétés `lines_before` et `lines_after`. À `false` ou 0, aucune ligne ne sera ajoutée.

On peut même mettre `line_after` à `-1` lorsqu'il arrive, parfois, qu'une ligne supplémentaire soit ajoutée par erreur (les calculs de Prawn sont parfois impénétrables). Avant les doubles colonnes suivantes, nous avons écrit le code : `(( columns(2, lines_before:0, space_after: -LINE_HEIGHT) ))` (nous avons volontairement utilisé la traduction anglaise « `columns` »).

Début du texte en double colonnes. Un  
texte en double colonnes qui ne devrait  
pas présenter de ligne vide de  
séparation ni au-dessus avec le texte

avant ni au-dessous avec le texte après.  
Tous ces textes devraient être collés. Fin  
du texte en double colonnes.

Paragraphe sous le texte en double colonnes collées. Ci-dessus, nous avons dû jouer sur `space_after`, avec une valeur négative, pour arriver à nos fins car `lines_after` restait inefficace. Au-dessus, on peut aussi jouer avec `space_before` si on veut définir l'espace avant. Notez que le texte est quand même remis sur des lignes de référence à chaque fois.

## Ligne en trop dans les multi-colonnes

Parfois il peut arriver que **Prawn-For-Book** compte une ligne de trop dans les colonnes, ce qui produit cet alignement pas très heureux :

Premier	Deuxième en regard
Deuxième	Troisième en regard
Troisième	
Premier en regard	

Pour palier cet écueil, on met la propriété `no_extra_line_height` à `true` dans la définition des colonnes. On obtient alors :

Premier	Premier en regard
Deuxième	Deuxième en regard
Troisième	Troisième en regard

---

# Le Mode Expert

## Description

Le *mode Expert* permet d'élargir de façon exponentielle les possibilités de **Prawn-For-Book** afin de produire les contenus les plus variés et les plus originaux.

Il demande une compétence particulière dans le langage Ruby ainsi qu'une bonne connaissance de la gem `Prawn` qui permet de produire les livres avec **Prawn-For-Book**.

## Indication du contexte d'erreur

À titre préventif dans les méthodes personnalisées, *helpers* et autres modules, on peut indiquer le contexte qui devra être affiché en cas d'erreur.

Cela se fait en utilisant le code `PFBError.context = "Le contexte"`.

```
\# ruby
def monHelper(pdf, book)
  12.times do |i|
    # Indiquer le contexte
    PFBError.context = <<~EOC
      Dans la boucle de calcul et d'écriture du
      chiffre, avec i = \#{i}
    EOC
    écrire_ce_chiffre(i)
  end
  # Penser à "défaire" le contexte
  PFBError.context = nil
end
```

## Injection

`book.inject(pdf, string, idx, self)` est vraiment la formule magique pour ajouter du contenu au livre. L'avantage principale de cette méthode est d'analyser précisément le type de contenu —représenté ici par `string`— et de le traiter conformément à son type. Par exemple :

- si `string` est `"![images/mon.svg]"`, alors ce sera une image qui sera traitée,
- si `string` est `"### Mon beau titre"` alors c'est un titre qui sera inséré,
- si `string` est `"(( new_page ))"` alors on passera à la nouvelle page<sup>1</sup>.

---

<sup>1</sup> Bien sûr, ici, dans le programme, on pourrait utiliser `pdf.start_new_page`, mais l'idée derrière l'utilisation de

`book.inject(...)` est de pouvoir utiliser le même code que dans le livre. Inutile d'apprendre une nouvelle langue ou de fouiller dans le code du programme pour savoir comment exécuter telle ou telle action.

---

`idx` correspond à l'index du paragraphe dans la source injectée, il n'a de valeur que pour le débogage. Dans le programme, il correspond par exemple au numéro de ligne dans le fichier. On pourra l'utiliser comme l'on veut.

`self` correspond dans le programme à l'instance du fichier de texte (`Prawn4book::InputTextFile`). On peut le définir comme tel si le code injecté vient d'un fichier (même si, dans ce cas, il vaudrait mieux utiliser tout simplement la `string`: (`include mon/fichier.md`)). Si elle n'est pas fournie, elle sera égale à `"user_metho"`.  
`{{TODO: Développer encore}}`

## Évaluation du code ruby

Tous les codes qui se trouveront entre « `#{...}` » (ou entre « `#{{{...}}}` » lorsque le code contient des accolades) seront évalués en tant que code ruby, dans le cadre du livre (c'est-à-dire qu'ils pourront faire appel à des méthodes personnalisées). Typiquement, on peut par exemple obtenir la date courante ou le numéro de version du livre pour l'insérer dans les premières pages à titre de repère.

## Évaluation au second tour

Certaines opérations ou certains calculs ne peuvent s'opérer qu'au second tour de l'application<sup>3</sup> — typiquement, le nombre de pages du livre —. On utilise alors la tournure suivante pour réaliser ces opérations.

```
#{{{ "#{operation>}" if Prawn4book.second_turn? }}}4
```

Dans le code ci-dessus, le contenu des guillemets ne sera évalué qu'au second tour de l'application. Mais attention, cela peut occasionner un changement des numéros de page si le texte ajouté au second tour est conséquent. Il est donc plus prudent de mettre au premier tour un texte d'environ la longueur du résultat attendu pour ne pas fausser le suivi. Pour le numéro des pages, que nous estimons au départ à plusieurs centaines mais moins d'un millier nous utilisons :

```
#{{{ Prawn4book.first_turn? ? « XXX » : « #{book.pages.count} » }}}
```

... qui signifie qu'au premier tour, **Prawn-For-Book** marquera simplement « XXX » et aux suivants il inscrira le nombre de pages.

---

<sup>3</sup> C'est le cas par exemple de l'impression du nombre de pages de ce manuel dans l'avant-propos, c'est-à-dire alors que le livre est à peine esquissé.

<sup>4</sup> Noter ici l'utilisation des trois accolades obligatoires lorsque le code lui-même a recours aux accolades.

*Si le fichier « `texte.pfb.md` » contient...*

---

Une opération simple permet de savoir que  $2 + 2$  est égal à `{2+2}` et que le jour courant (au moment de l'impression de ce livre) est le `{Time.now.strftime('%d %m %Y')}`.

*Le livre final (document PDF) contiendra :*

---

Une opération simple permet de savoir que  $2 + 2$  est égal à 4 et que le jour courant (au moment de l'impression de ce livre) est le 21 12 2023.

---

## Bibliographies en mode expert

TODO: Décrire comment faire une méthode de formatage propre dans `Prawn4book::Bibliography` (méthode d'instance) pour l'utiliser dans la liste des sources, pour une propriété particulières. Si, par exemple, la donnée `format` de la bibliographie (dans la recette), définit `{title|mon_transformeur}`, alors il faut définir la méthode `Prawn4book::Bibliography#mon_tranformeur` qui reçoit en argument la valeur de `:title` de l'item.

TODO: Montrer qu'on peut par exemple définir une font et/ou une couleur propre comme pour la bibliographie `costume`

## Mode Multi-colonnes

En tant qu'expert, vous pouvez utiliser le mode multi-colonnes (affichage du texte sur plusieurs colonnes) à l'intérieur des formateurs.

Bien entendu, vous pouvez, si vous êtes parfaitement à l'aise avec ça, utiliser les `column_box` de **Prawn**. Mais **Prawn-For-Book** propose là aussi des outils plus élaborés qui permettront une mise en page assistée.

Imaginons l'index `entree` qui permet de gérer les entrées du dictionnaire que vous construisez. À la fin du livre, vous voulez afficher cette index sur trois colonnes. Vous allez donc implémenter la méthode `CustomIndexModule#print_index_entree` dans le fichier `formater.rb`.

Elle ressemblera au code de la page suivante.



```
module CustomIndexModule
```

```

# Méthode appelée automatiquement si le code
# `(( index\\(entree) ))`
# est utilisé dans le texte
#
def print_index_entree(pdf)

  # Les paramètres d'instanciation du multi colonnes
  params = {
    column_count: <nombre de colonnes>,
    width:        <largeur si autre que page complète>,
    gutter:       <gouttière si autre que valeur défaut>,
    lines_before: <nombre de lignes avant si autre que 1>,
    lines_after:  <nombre de lignes après si autre que 1>,
    space_before: <espace avant si nécessaire>,
    space_after:  <espace après si nécessaire>
  }
  # On instancie un texte multi-colonnes
  mc_block = Prawn4book::PdfBook::ColumnsBox.new(book, **params)
  # items contient la liste de toutes les entrées
  items.each_with_index do |item_id, occurrences, idx|
    # On traite les items pour obtenir le texte
    str = ...
    # On en fait une instance de paragraphe
    para = Prawn4book::PdfBook::NTextParagraph.new(
      book: book,
      raw_text: str,
      pindex:idx,
    )
    # On peut indiquer la source, pour les messages d'erreur
    # éventuels
    para.source = "Construction de l'index des entrées"
    # On peut supprimer l'indentation éventuelle
    para.indentation = 0
    # Et on injecte le paragraphe dans le bloc multi-colonnes
    mc_block << para
  end
  # On diminue la fonte à utiliser et on l'utilise
  fonte = Prawn4book::Fonte.dup_default
  fonte.size = 10
  pdf.font(fonte)
  # Il suffit maintenant d'imprimer ce bloc multi-colonnes
  mc_block.print(pdf)
  # Done!
end
end #/module
```



# Annexe

## Reconstruction du manuel

Pour voir certaines fonctionnalités, il est nécessaire de relancer la construction de ce manuel autoproduit. Pour ce faire, suivez les étapes suivantes :

- dans une fenêtre Terminal, jouez la commande `pfb open`,
- dans la liste, choisir « Ouvrir le dossier du manuel »,
- ouvrir un Terminal dans ce dossier (control-clic sur le dossier puis choisir « Nouveau Terminal au dossier » ou similaire),
- jouer la commande `pfb build` dans ce Terminal (parfois il pourrait être demandé d'ajouter l'option de débogage, dans ce cas il faudra jouer `pfb build -debug`).

Le manuel autoproduit se reconstruit alors en quelques secondes.

## Marques markdown

Vous trouverez ci-dessous toutes les marques Markdown utilisables.

*Si le fichier « `texte.pfb.md` » contient...*

---

La table ci-dessous a été construite aussi en markdown, en utilisant le format :

```
| CA1 | CA2 | CA3 | CA4 |  
| CB1 | CB2 | CB3 | CB4 |  
|/|
```

Vous pouvez trouver toutes les informations sur l'utilisation des tables à la page [### REF: tables ###](#).

*Le livre final (document PDF) contiendra :*

---

---

italique	<code>* ... *</code>	<i>texte en italique</i>
----------	----------------------	--------------------------

---

gras	<code>** ... **</code>	<b>texte en gras</b>
------	------------------------	----------------------

---

souligné	<code>~ ... ~</code>	<u>texte souligné</u>
----------	----------------------	-----------------------

---

## Le format YAML de la recette

exposants 1<sup>er</sup> 1<sup>er</sup>  
Le format **YAML** est un format très simple de présentation et de consignation des données. Il est utilisé dans **Prawn-For-Book** pour définir recette/juste\_titre (p. ), que ce soit pour un titre unique (cf. page Recette du livre (p. 21)) ou pour une collection (cf. page Recette de la collection (p. 21)). 2<sup>e</sup> 2<sup>e</sup>

Les données sont *imbriquées*, comme nous l'avons dit, pour s'y retrouver plus facilement, entendu que les recettes peuvent contenir de nombreuses informations. Voyez l'imbrication donnée en exemple ci-dessous. X<sup>1e</sup> X<sup>e</sup>

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```
---  
table_de_donnees:  
  sous_ensemble_liste:
```

```

- premier item
- deuxième item
sous_ensemble_texte: "Ma donnée texte"
un_nombre: 12
une_date: "2023-11-22"
quelquun:
  prenom: "Marion"
  nom: "MICHEL"

```

---

## Les "Fontes-Strings"

Pour définir les polices dans les éléments, à commencer par la recette, on utilise de préférence ce qu'on appelle dans **Prawn-For-Book** les « **fonte-string** » (« string » signifie quelque chose comme « caractère » en anglais).

Ces **fonte-strings** se présentent toujours de la même manière, par une chaîne de caractères (de lettres) contenant 4 valeurs séparées par des balances, dans l'ordre :

- le **nom** de la police,
- le **style** de la police (qui doit être définie,
- la **taille** à appliquer au texte (en points-pdf),
- la **couleur** éventuelle (pour la définition de la couleur, voir Définition de la couleur (p. 37)).

*Si le fichier `recipe.yaml` ou `recipe_collection.yaml` contient...*

---

```

---
font: "<police>/<style>/<taille>/<couleur>"
# Par exemple
font: "Numito/bold_italic/23/FF0000"

```

---

## Définition de la couleur

En règle générale, la couleur dans **Prawn-For-Book** peut se définir de deux façons :

- en hexadécimal (comme en HTML) à l'aide de 6 chiffres/lettres hexadécimal (donc de 0 à F),
- en quadrichromie, avec CMJN (Cyan, Magenta, Jaune, Noir).

### COULEUR HEXADÉCIMALE

En hexadécimal, on a 3 paires de deux chiffres/lettres qui représentent respectivement les quantités de rouge, vert et bleu. Par exemple « AAD405 » signifiera « AA » de rouge, « D4 » de vert et « 05 » de bleu.

"000000" représente le noir complet, « FFFFFFFF » représente le blanc complet. Lorsque toutes les valeurs sont identiques (par exemple « CCCCCC ») on obtient un gris (mais il existe d'autres moyens d'obtenir du gris).

Quelques couleurs hexadécimales aléatoires :

- Couleur dd2b80
- Couleur 7a2046
- Couleur 7f44e3
- Couleur c5febe
- Couleur 66ebc8
- Couleur ee378f
- Couleur 83fcac
- Couleur e0f54c
- Couleur ebb10c
- Couleur e2ed26

## COULEUR QUADRICHROMIQUE

La quadrichromie, représentée souvent par « CMJN » (ou « CMYK » en anglais), est le format de la couleur en imprimerie. On aura une liste (crochets) contenant les 4 valeurs de 0 à 127 pour le Cyan (C), le Magenta (M), le Jaune (Y) et le noir (K). Par exemple « [0, 12, 45, 124] » signifiera qu'il n'y aura pas de Cyan, qu'il y aura 12 de magenta, 45 de jaune et 124 de noir. [0, 0, 0, 0] représente le blanc complet, [127, 127, 127, 127] le noir complet.

Quelques couleurs quadrichromiques :

- Couleur [0,0,0,127]
- Couleur [127,0,0,127]
- Couleur [127,0,0,0]
- Couleur [0,127,0,127]
- Couleur [0,127,0,0]
- Couleur [0,0,127,127]
- Couleur [0,0,127,0]
- Couleur [127,127,127,127]
- Couleur [127,127,127,0]
- Couleur [70, 65, 100, 60]
- Couleur [15, 36, 7, 76]
- Couleur [12, 2, 34, 3]
- Couleur [72, 58, 69, 10]
- Couleur [104, 106, 13, 79]
- Couleur [39, 16, 6, 37]
- Couleur [108, 73, 101, 62]
- Couleur [39, 27, 69, 62]
- Couleur [96, 51, 50, 48]
- Couleur [93, 108, 14, 101]

## Outils couleurs *online*

Vous trouverez sur le net, en tapant simplement « Couleur HTML CMJN » dans un moteur de recherche, des outils *en ligne* vous permettant de choisir des couleurs précises et d'obtenir leur code.

## Rogner une image SVG

Il est fort possible qu'en produisant une image SVG, et en l'insérant dans le livre, elle laisse voir trop de blanc autour d'elle, comme dans le premier exemple donné ci-après. Pour palier ce problème, il faut « rogner » cette image SVG. Mais *rogner* une image SVG ne se fait pas aussi facilement qu'avec une image d'un format non vectoriel (JPG, PNG, etc.). Il faut pour ce faire utiliser, après avoir chargé la commande `inkscape` dans votre ordinateur, le code suivant :

```
inkscape -l -D -o image-rogned.svg image.svg
```

Le livre final (document PDF) contiendra :

---

Image non rognée :







(note : la page vide précédente est occupée par le blanc de l'image rognée)

La même image, rognée cette fois :



Amet exercitation ut dolore in in nulla adipisicing laborum.

---

## Synopsis de création

Cette section présente une sorte de synopsis que vous pouvez suivre pour créer votre livre.

- ...
- ajustez parfaitement vos images en jouant sur leur `adjust` :. N'hésitez pas à ajouter des lignes avec ( `line` ) pour bien séparer les choses si nécessaire
- ...

## Installation de l'application

Pour que l'installation de l'application fonctionne, il faut avoir une version ruby « up and running » sur son ordinateur et avoir chargé le dossier principal dans son dossier `/Applications`.

Il faut créer ensuite un alias `pfb` qui lancera l'application, grâce à la commande :

### Sur MacOS

```
> ln -s /Application/Prawn4book/prawn4book.rb /usr/local/bin/pfb
```

### Sur Windows

{{TODO}}

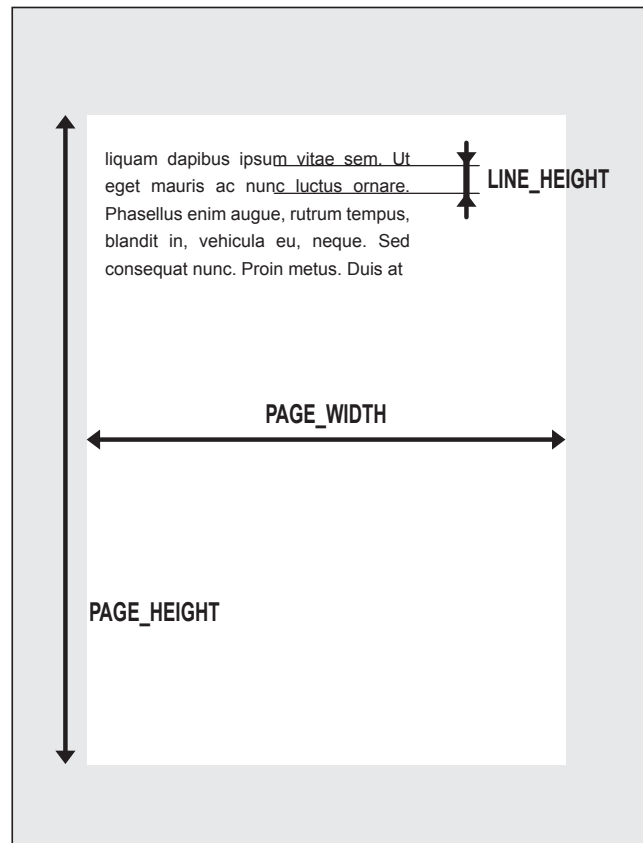
## Constantes

Trouvez ci-dessous la liste des constantes utilisables dans les paramètres des méthodes.

- **LINE\_HEIGHT** | (« Hauteur de ligne » en anglais) Hauteur courante de la ligne.
- **PAGE\_HEIGHT** | (« Hauteur de page » en anglais) Hauteur de la page du livre hors marge (c'est-à-dire qu'on ne compte pas les marges dans cette longueur, c'est la hauteur à l'intérieur des marges).

- **PAGE\_WIDTH** | (« Largeur de page » en anglais) Largeur de la page du livre hors marge.

Vous pouvez retrouver toutes ces informations dans l'image ci-dessous.



## Liste exhaustive des fonctionnalités

- définition de la taille du livre
- pagination automatique et personnalisable
- définition de la fonte par défaut
- définition des pages spéciales à insérer pages\_speciales/pages\_speciales (p. )
- justification par défaut des textes
- colorisation des textes
- suppression des veuves et des orphelines
- suppression automatique des lignes de voleur
- nombreuses sortes de puces et puces personnalisées puces/puces (p. )
- évaluation à la volée du code ruby (pour des opérations, des constantes, etc.)
- traitement des références croisées
- traitement dynamique des références à d'autres livres
- gestion par défaut ou personnalisée des images et de leur légende
- génération dynamique de contenu
- corrige l'erreur typographique de l'apostrophe droit
- corrige l'erreur typographique de l'absence d'espace avant et après les chevrons

- corrige l'erreur typographique de l'espace avant et après les guillemets droits et courbes
- corrige l'oubli de l'espace avant les ponctuations doubles
- corrige l'erreur d'espace avant les ponctuations doubles (pose d'une insécable)
- corrige l'absence d'espace insécable à l'intérieur des tirets d'exergue
- changement de fonte (police) pour le paragraphe suivant  
change\_fonte\_for\_next\_paragraph (p. )
- placement sur n'importe quelle ligne de la page
- exportation seulement du texte produit
- exportation comme livre numérique (pur PDF)



AVANT-PROPOS .....	9
FONCTIONNALITÉS .....	13
Généralités .....	15
Aperçu rapide des commandes .....	15
Les Forces de <i>Prawn-For-Book</i> .....	15
Définitions minimales .....	17
Définition des marges .....	18
Les Recettes .....	21
Recette du livre .....	21
Recette de la collection .....	21
Définition de la fonte par défaut .....	22
Éditeur / Maison d'édition .....	22
Aides & Assistants .....	23
Manuel et autres aides .....	23
Les « Snippets » .....	24
Messages d'erreurs et notices .....	25
Exportation du texte .....	25
Format du livre et des pages .....	27
Page en multi-colonnes .....	27
Le Mode Expert .....	30
Description .....	30
Indication du contexte d'erreur .....	30
Injection .....	30
Évaluation du code ruby .....	31
Bibliographies en mode expert .....	32
Mode Multi-colonnes .....	32
ANNEXE .....	35
Reconstruction du manuel .....	36
Marques markdown .....	36
Le format YAML de la recette .....	36
Les "Fontes-Strings" .....	37
Définition de la couleur .....	37
Rogner une image SVG .....	39
Synopsis de création .....	42

Installation de l'application .....	42
Constantes .....	42
Liste exhaustive des fonctionnalités .....	43







Icare Éditions

Conception  
Philippe Perret  
([philippe.perret@icare-editions.fr](mailto:philippe.perret@icare-editions.fr))

Mise en page  
Prawn-for-book

Couverture  
MM & PP

Correction & relecture  
Marion Michel

Imprimé par Prawn-For-Book