# Prawn4book
# Manuel


**Prawn4book<br />Manuel**

- Introduction
  - Présentation
  - Les grandes forces de Prawn-for-book
  - Commande(s)
- Obtenir de l'aide
- Création d'un livre
- Création d'une collection
- Ajouter un livre à une collection
- Construction du PDF du livre
  - Ouvrir le fichier PDF produit
  - Options de fabrication (pour le travail)
    - Affichage des marges
    - Affichage de la grille de référence
    - Affichage d'un rang précis de pages
    - Affichage du curseur
- Ouverture du PDF
- Texte du livre
  - Package Sublime Text
  - Modifier l'aspect du texte dans Sublime Text (son affichage dans l'application)
- Le livre pour l'impression
  - Les marges
  - Pagination
  - Les titres
    - Grand titre sur une belle page
    - Exclure un titre de la table des matières
  - Les paragraphes
    - Définition
    - Les différents types de paragraphe
    - Paragraphes de texte
    - Titres
    - Images
    - Les paragraphes-codes (pfb-code)
    - Numérotation des paragraphes
  - Saut de page
  - Insertion d'un texte externe
  - Headers & Footers (entêtes et pieds de page)
    - Principe
    - Rangs de pages
    - Contenu
    - Définition des entêtes et pieds de page
    - Positionnement

- Tiers et contenus
- Dans la recette
- Variables
- Pages spéciales
- Page de titre
- Page d’informations
- Table des matières
- Page d'index
- Pages de bibliographie
- La balise de la bibliographie
- Le titre de la bibliographie
- Chemin d’accès aux données de la bibliographie
- La page de la bibliographie
- Les données de la bibliographie
- Mise en forme des données bibliographiques
- Références (et références croisées)
- Références croisées
- Référence croisée vers un livre non prawn
- Fichier de références
- Exclure des paragraphes
- Commentaires
- Méthodes de traitement et de formatage propres
- Méthode d’helpers — `(( #<method>(<args>) ))`
- Formatage personnalisé (`formater.rb`)
- Formatage des paragraphes
- Formatage des éléments de bibliographie
- Parsing personnalisé des paragraphes (`parser.rb`)

Recette du livre ou de la collection

- Définition
- Création de la recette du livre
- Contenu de la recette du livre
- Informations générales
- Informations générales pour une collection
- Aspect général du livre
- Aspect des pages
- Données des titres
- Données de l'éditeur/éditions
- Fontes
- Tous les types de page

Annexe

- Points PDF

Ne pas afficher les espaces insécables

Package Sublime Text

- Choix d'une autre police

Prawn

- Blocs de texte avec Prawn

# Introduction

## Présentation

**Prawn4book** — ou **Prawn For Book**, c'est-à-dire « Prawn pour les livres » — est une application en ligne de commande permettant de transformer un simple texte en véritable PDF prêt pour l'impression, grâce au (lovely) gem `Prawn` (d'où le nom de l'application.

L'application met en forme le texte, dans ses moindres détails et ses moindres aspects, empaquette les polices nécessaires, gère les références — même les références croisées —, gère les index et les bibliographies — autant que l'on veut — et produit un PDF conforme en tout points à ses désirs.

## Les grandes forces de Prawn-for-book

Les grandes forces de ***PRAWN-FOR-BOOK*** sont donc :

- mise en forme du texte dans ses moindre détails (feuilles de style, modules complexes — experts — de formatage),
- gestion des références internes (renvois, références à une page ou un paragraphes, etc.),
- gestion des références croisées (références à la page d'un autre livre)
- gestion d'un index,
- gestion d'autant de bibliographies que l'on veut,
- gestion automatiquement de la table des matières (est-ce vraiment utile de le préciser ?…)

## Commande(s)

Sa commande simple est (*) :

```
$> pfb
```

Ou en version longue (*) :

```
$> prawn-for-book
```

> (*) En présupposant bien sûr que des alias de commande ont été créé, sur MacOs grâce à :
>
> ```
> ln -s /Users/me/Programmes/Prawn4book/prawn4book.rb /usr/local/bin/prawn-for-book
> ln -s /Users/me/Programmes/Prawn4book/prawn4book.rb /usr/local/bin/pfb
> ```
>
> Et sur Windows grâce à :
>
> TODO ?

---

# Obtenir de l'aide

On peut obtenir de l'aide de différents moyens :

- `$> pfb aide` ouvrir une aide générale en présentant les commandes principales.
- `$> pfb aide <identifiant>` offrira de l'aide sur l'`<identifiant>`. On peut obtenir grâce à

cette commande les assistants de création qui permettent de définir très précisément la recette d'un livre ou d'une collection.
- `$> pfb lexique "groupe de mots"` offrira de l'aide sur un mot particulier ou un groupe de mots en en donnant la définition ou le sens dans *Prawn-for-book*. Note : les guillemets ne sont nécessaires que s'il y a plusieurs mots.

---

# Création d'un livre

- Choisir le dossier dans lequel doit être créé le livre,
- ouvrir une fenêtre Terminal dans ce dossier,
- jouer la commande `$> pfb init`,
- choisir de construire un nouveau livre,
- suivre l'assistant pour définir les données du livre (ou n'en définissez aucune, vous aurez toujours le loisir de le faire plus tard).

# Création d'une collection

Avec **Prawn-for-book**, on peut aussi créer des collections, c'est-à-dire un ensemble de livres qui partageront les mêmes éléments, à commencer par la charte graphique. Plutôt que d'avoir à la copier-coller de livre en livre, entrainant des opérations lourdes à chaque changement, on crée une collection qui définira les éléments communs et on met les livres dedans.

- Choisir le dossier dans lequel doit être créée la collection,
- ouvrir une fenêtre Terminal à ce dossier,
- jouer la commande `$> pfb init`,
- choisir de construire une collection,
- suivre l'assistant de création.

---

# Ajouter un livre à une collection

Suivre la procédure d'initiation d'un nouveau livre mais en ouvrant le Terminal au dossier de la collection (ou au dossier du livre créé dans le dossier de cette collection).

---

# Construction du PDF du livre

Pour lancer la fabrication du PDF qui servira à l'impression du livre, jouer la commande :

```
> cd path/to/book/folder
> pfb build
```

**À bien noter : cette commande fabrique vraiment le PDF qu'il suffira d'envoyer à l'imprimeur pour tirer le livre.**

# Ouvrir le fichier PDF produit

Pour ouvrir le document PDF à la fin de la fabrication, ajouter l'option `--open`.

```
$> prawn-for-book build --open
```

# Options de fabrication (pour le travail)

Certaines options permettent de travailler le livre avant sa fabrication définitive. On peut par exemple :

- demander l'affichage des marges,
- demander l'affichage de la grilles de référence (la grille sur laquelle se calent les lignes pour être bien alignées),
- demander la fabrication de seulement quelques pages, voire une seule,
- l'affichage de la hauteur du curseur.

## Affichage des marges

On peut par exemple demander l'affichage des marges à l'aide de l'option **`--display_margins`** au moment de la fabrication du livre :

```
$> pfb build -display_margins
```

Utiliser le paramètre `grid` pour préciser les pages sur lesquelles doivent être dessinées les marges (sans cette précision elles seront dessinées sur toutes les pages) en les séparant d'un tiret simple. Par exemple :

```
$> pfb build -display_margins grid=4-12
```

… pour n'afficher les marges que sur les pages de 4 à 12.

## Affichage de la grille de référence

On peut afficher les lignes de la grille de référence (pour voir comment seront alignées les lignes du texte) à l'aide de l'option **`--display_grid`** au moment de la fabrication du livre :

```
$> pfb build -display_grid
```
ou
```
$> pfb build -g
```

Utiliser le paramètre `grid` pour préciser les pages sur lesquelles doivent être dessinées les lignes de références (sans cette précision elles seront dessinées sur toutes les pages) en les séparant d'un tiret simple. Par exemple :

```
$> pfb build -display_grid grid=4-12
```

… pour n'afficher la grille de référence que sur les pages de 4 à 12.

## Affichage d'un rang précis de pages

Pour commencer à une page précise, utiliser l'option simple `-first` avec le numéro de la page :

```
$> pfb build -first=12
```

Pour s'arrêter à une page précise, par exemple la 4$^{e}$, utiliser l'option simple `-last` avec le numéro de page :

```
$> pfb build -last=4
```

Un usage très utile, par exemple, si l'on est limité à un nombre minimal de pages comme sur KDP (24) mais qu'on ne veut pas imprimer tout le livre (s'il est gros) consiste à sortir le PDF avec seulement les 24 premières pages et d'envoyer le PDF pour impression.

```
$> pfb build -last=24
```

### Affichage du curseur

Avec l'option `-c/--cursor` on peut demander à ce que les positions curseur soient ajoutées au livre.

---

# Ouverture du PDF

On peut ouvrir le PDF du livre dans Aperçu à l'aide de la commande :

```
$> pfb open book
```

---

# Texte du livre

On peut travailler le texte du livre dans n'importe quel éditeur simple. Sublime Text est mon premier choix pour le moment. Notamment parce qu'il offre tous les avantages des éditeurs de code, à commencer par l'édition puissante et la colorisation syntaxique. Il suffit que le texte se termine par **`.pfb.txt`** ou **`.pfb.md`** pour que Sublime Text applique le format *Prawn4Book*.

## Package Sublime Text

Ce package est défini dans le dossier package `Prawn4Book` de Sublime Text. On peut ouvrir ce package rapidement en jouant :

```
$> prawn-for-book open package-st
```

## Modifier l'aspect du texte dans Sublime Text (son affichage dans l'application)

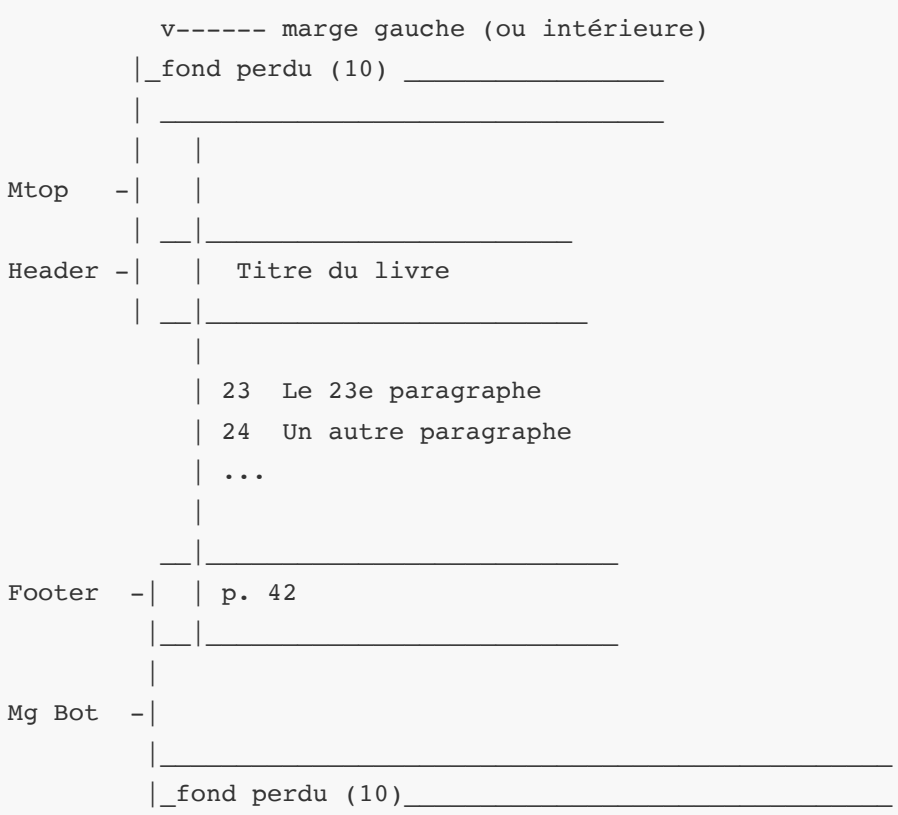
Pour modifier l'aspect du texte, il faut ouvrir le package dans *Sublime Text* (`$> prawn-for-book open package-st`) et modifier le code dans le fichier `Prawn4Book.sublime-settings` (pour la police, la taille de police, etc.) ou le fichier `Prawn4Book.sublime-color-scheme` (pour modifier la colorisation syntaxique ou les scopes).

---

# Le livre pour l'impression

## Les marges

Les marges sont définies de façon très strictes et concernent vraiment la partie de la page ***où ne sera rien écrit***, ni pied de page ni entête. On peut représenter les choses ainsi :




Ce qui signifie que le haut et le bas du texte sont calculés en fonction des marges et des header et footer.

> Noter qu'il y a toujours un fond perdu de 10 post-script points autour de la page.

---

## Pagination

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | `book_format:page:` | **:numerotation** | pages/parags |

Une des grandes fonctionnalités de *Prawn-for-book* est de permettre de paginer de deux manières :

- à l'aide des numéros de pages (pagination traditionnelle),
- à l'aide des numéros de paragraphes (pagination "technique" permettant de faire référence à un paragraphe précis, par son numéro/indice).

> La numérotation des paragraphes peut être très pratique aussi quand on veut recevoir des commentaires précis — et localisés — sur son roman ou tout autre livre. Vous pouvez l'utiliser pour le PDF que vous remettez à vos lecteurs et lectrices.

Pour se faire, on règle la valeur de la propriété **`book_format:page:numerotation`** dans la recette du livre ou de la collection. Les deux valeurs possibles sont `pages` (numérotation des pages) ou `parags` (numérotation des paragraphes).

> Modifier la valeur directement dans le fichier recette du livre ou de la collection nécessite une certaine habitude. Il est préférable, pour tous les réglages, de passer par les assistants. Ici, il suffit de jouer `$> pfb assistant` et de choisir “Assistant format du livre”, puis de renseigner la propriété “Numérotation”.

Cette valeur influence de nombreux éléments du livre, dont :

- les numéros en bas de page (si on les désire)
- les index
- les repères bibliographiques
- les marques de références

Pour savoir comment placer et formater les numéros de pages, cf. Headers et Footers.

---

# Les titres

La base du texte étant du markdown, les titres s”indiquent avec des dièses en fonction de leur niveau :

```
# Grand titre
## Titre de chapitre
### Titre de sous-chapitre
#### Titre de section
etc. si nécessaire.
```

Pour la mise en forme des titres dans le livre, voir la définition des titres dans la recette du livre.

## Grand titre sur une belle page

Pour qu'un grand titre se retrouve toujours sur une belle page (ie la page impaire, à gauche), on doit mettre sa propriété `:belle_page` à `true` dans la recette du livre ou de la collection.

Pour la mise en forme des titres dans le livre, voir les titres dans la recette du livre.

## Exclure un titre de la table des matières

Pour exclure un titre de la table des matières, c’est-à-dire pour qu’il soit inscrit en tant que titre dans le texte mais qu’il n’apparaissent pas dans la table des matières, il suffit de mettre `{no-tdm}` dans ce titre, n’importe où sauf avant les dièses. Par exemple :

```
# {no-tdm} Titre exclus de la tdm

# Titre dans la tdm

## Autre titre exclus {no-tdm}
```

---

# Les paragraphes

## Définition

L'unité textuel de *Prawn-for-book* est le paragraphe (mais ce n'est pas l'atome puisqu'on peut introduire des éléments sémantiques dans le paragraphe lui-même, qui seront évalués "en ligne").

## Les différents types de paragraphe

- les Paragraphes de texte,
- les Titres,
- les Images,
- les Pfb-codes.

## Paragraphes de texte

Le paragraphe de texte se définit simplement en l'écrivant dans le fichier `.pfb.md`.

```
Définit dans le texte par un texte ne répondant pas aux critères suivants. Un
paragraphe peut commencer par autant de balises que nécessaire pour spécifier les
choses. Par exemple :
citation::bold::center:: Une citation qui doit être centrée.
```

Il existe ensuite plusieurs manières de styliser ces paragraphes si nécessaire :

- stylisation par défaut,
- stylisation en ligne de portion de textes dans le paragraphe,
- stylisation *inline* (en ligne),
- stylisation par balise initiale.

#### STYLE PAR DÉFAUT DU PARAGRAPHE

| | Recette | propriété | valeurs possibles | |
|---|---|---|---|---|
| | | **:default_font_n_style:** | Nom de fonte (police) chargée et le style utilisé | “Garamond/italic” |
| | | **:default_font_size:** | Nombre entier ou flottant | 12.4 |
| | | **:default_font_style** | [OBSOLÈTE] Un des styles défini pour la fonte | |

On définit le style du paragraphe par défaut dans la recette du livre ou de la collection en définissant les propriétés `:default_font` (nom de la fonte, qui doit être chargé dans le document), `:default_font_size` (taille de la police) et `:default_font_style` (style défini pour la fonte, en général ‘:nomal’.

#### STYLE DE PORTIONS DE TEXTES DANS LE PARAGRAPHE

Le paragraphe peut contenir de la mise en forme simple, "en ligne", comme le gras ou l'italique, en entourant les mots avec `<i>...</i>` ou `<b>...</b>`. Par exemple :

```
Un mot en <b>gras</b> et un mot en <i>italique</i>. Une expression en <i><b>gras et
italique</b></i>.
```

#### STYLISATION "INLINE" DU PARAGRAPHE — `(( {<hash} ))`

Un paragraphe peut être complètement modifié en utilisant ce qu'on appelle la *stylisation inline* qui consiste à ajouter une ligne juste au-dessus du paragraphe qui contient ses propriétés modifiées. Par exemple :

```
Un paragraphe au style par défaut.

(( {<data>} ))
Le paragraphe influencé par les <data> ci-dessus.
```

> Noter les `(( ... ))` (doubles-parenthèses) qui sont la marque de Prawn-for-book et les crochets qui vont définir une table de propriété (un *dictionnaire*, comme dans un langage de programmation.

On peut, à la base, changer par exemple la taille du texte pour ce paragraphe avec la propriété `:font_size`.

```
(( {font_size:22} ))
Ce paragraphe aura une taille de 22 pour la police courante.
```

La propriété `font_family` permet de changer de fonte (à nouveau il faut que cette fonte soit accessible).

```
(( {font_family: "Arial"} ))
Ce paragraphe sera en Arial, dans la taille par défaut de la police par défaut.
```

On peut mettre plusieurs propriétés en les séparant par des virgules :

```
(( {margin_left: 40, margin_top: 50} ))
IMAGE[images/mon_image.svg]
```

L'image ci-dessus se retrouvera à 40 points-pdf de la marge gauche et à 50 points-pdf de son contenu précédent.

Les propriétés qu'on peut définir sont les suivantes :

| Propriété | Description | Valeurs |
|---|---|---|
| **font_family** | Nom de la fonte (qui doit exister dans le document) | String (chaine), par exemple `font_famiily:"Garamond"` |
| **font_size** | Taille de la police | Entier ou valeurs. P.e. `font_size:12` |
| **font_style** | Style de la police à utiliser (doit être défini pour la police) | Symbol (mot commençant par ":"). P.e. : `font_style: :italic` |
| **kerning** | Éloignement des lettres | Entier ou flottant. P.e. `kernel:2` |
| **word_space** | Espacement entre les mots | Entier ou flottant. P.e. `word_space: 1.6` |
| **margin_top** | Distance avec l'élément au-dessus | Entier en points-pdf ou valeur. P.e. `margin-top: 2.mm` |
| **margin_right** | Distance avec la marge droite | Idem |
| **margin_bottom** | Distance avec l'élément inférieur | Idem |
| **margin_left** | Distance de la marge gauche | Idem |
| **width** | Largeur de l'image (si c'est une image) | Pourcentage ou valeur avec unité. P.e. `width: "100%"` ou `width: 3.cm` (notez qu'il n'y pas de guillemets lorsqu'on utilise les unités Prawn. |
| **height** | Pour une image, la hauteur qu'elle doit faire. | |

### AJUSTEMENT DU PARAGRAPHE

Une propriété particulièrement utile pour de l'impression professionnelle concerne l'espacement entre les mots qui permet d'éviter les mots seuls en fin de paragraphes par exemple. Supprimer deux ou trois mots sur la dernière ligne peut permettre par exemple de faire remonter un titre de façon élégante.

Pour gérer cette fonctionnalité, on utilise la commande `(( del_last_line ))` ("delete the last line", "supprimer la dernière ligne"). L'application joue alors elle-même sur l'espacement entre les mots (voire entre les lettres) pour condenser un peu le texte.

**Par mesure de prudence**, pour obtenir un rendu acceptable, n'appliquez jamais cette commande s'il y a trop de mots sur la ligne à supprimer et/ou si le paragraphe est trop cours. Un paragraphe de moins de 4 lignes se met en danger si on lui applique cette commande.

Exemple d'utilisation :

```
(( del_last_line ))
Ceci  est  un texte  assez long  qui  doit  être  condensé
pour  que  sa  dernière  ligne  soit supprimée, en  jouant
sur  les espacements  entre chaque  mot, en les rapprochant
de  façon discrète  pour  que les trois derniers mots soient
rayés de la carte.
```

Après traitement, le paragraphe ressemblera à :

```
Ceci est un texte assez long qui doit être condensé pour que
sa dernière ligne soit supprimée, en jouant sur les espace-
ments entre chaque mot, en les rapprochant de façon discrète
pour que les trois derniers mots soient rayés de la carte.
```

Bien entendu, cette commande ne se place dans le texte du livre que lorsque le PDF a été construit et qu'on a constaté l'état du paragraphe. On ne peut pas le faire au hasard, il faut le faire comme le ferait un metteur en page, sur pièce.

**STYLISATION DU PARAGRAPHE PAR BALISE INITIALE**

Un paragraphe de texte peut également commencer par une *balise* qui va déterminer son apparence, son *style* comme dans une feuille de styles. Ces balises peuvent être communes (propres à l"application) ou personnalisées.

**Personnalisation des paragraphes texte (style de paragraphe personnalisés**

Les *styles de paragraphes personnalisés* doivent être identifiés par une *balise* qui sera placée au début du paragraphe à stylisé. Par exemple, si ma balise est `gros`, cela donnera :

```
gros::Le paragraphe qui sera mis dans le style personnalisé "gros".
```

Ensuite, pour fonctionner, il faut dire à *Prawn-for-book* comment styliser ce paragraphe.

Il existe deux manières de le faire :

- la manière simple, en ne se servant que des propriétés ci-dessus. Dans cette utilisation, le style permet simplement de ne pas avoir à répéter toute la ligne de définition du paragraphe avant le paragraphe.

  Pour cette manière, il faut définir dans le module **`FormaterParagraphModule`** du fichier `formater.rb` la méthode **`<balise>_formater(paragraph)`** qui reçoit en premier paramètre l'instance du paragraphe. Ensuite, à l'intérieur de cette méthode, on définit toutes les valeurs :

  ```
  module FormaterParagraphModule
    def formate_gros(par)
      par.font = "Arial"
      par.font_size = 14
      par.margin_left = "10%"
      par.kerning = 1.2
      par.margin_top = 4
      par.margin_bottom = 12
      par.text = "FIXED: #{par.text)"
    end
  end
  ```

  ou :

```
module FormaterParagraphModule
  def formate_gros(par)
    par.instance_eval do
      font = "Arial"
      font_size = 14
      # ...
      text = "FIXED: #{text}"
    end
  end
end
```

- la manière complexe, permettant une gestion extrêmement fine de l'affichage, mais nécessitant une connaissance précise de Prawn. Elle consiste à définir dans le module `FormaterParagraphModule` du fichier `formater.rb` la méthode **buildparagraph(paragraph, pdf)** qui reçoit en premier argument l'instance du paragraphe et en second argument l'instance `Prawn::View` du constructeur du livre. Ensuite, à l"intérieur de la méthode, on construit le paragraphe. Par exemple :

```
module FormaterParagraphModule
  def build_gros_paragraph(par, pdf)
    pdf.update do
      font(par.font, size: par.font_size)
      bounding_box([100, cursor], width: bounds.width/2, height: 100) do
        transparent(0.5) { stroke_bounds }
        image icone_tip, at: [...]
        text par.text,
      end
    end
  end
end
```

**Styles paragraphes texte commun**

| Balise | Description | Exemples |
|---|---|---|
| **dict::entry::** [TODO] | Entrée de dictionnaire | |
| **dict::text::** [TODO] | Description de l'entrée, le texte suivant l'entrée. | |
| | | |

## Titres

Le titre se définit comme en markdown c'est-à-dire à l'aide de dièses.

```
# Un grand titre
## Un chapitre
### Un sous-chapitre
etc.
```

## Images

Les images se définissent à l'aide de la balise :

```
IMAGE[<data>]
```

Les données sont composées d'un chemin d'accès à l'image, puis de données qui définissent l'image. Le **chemin d'accès** doit être soit absolu soit relatif.

> Tip : Il est préférence de mettre les images dans un dossier `images` se trouvant dans le dossier du livre ou de la collection et d'y faire référence simplement par `images/mon_image.jpg`.

Les images peuvent être de tout format, mais puisqu'elles sont destinées à l'impression, leur espace colorimétrique doit être le [modèle colorimétrique CMJN (Cyan, Magenta, Jaune, Noir)].

```
Ci-dessous une image qui sera présentée sur toute la largeur de la page (hors-marge).

IMAGE[images/pour_voir.jpg]
L'image gardera de l'air avant ce texte, même s'il est collé dans le texte.

Une image qui sera réduite de moitié.

IMAGE[images/red.jpg|width:50%]
```

### Propriétés de l'image

Trouvez ci-dessous la liste des propriétés qui peuvent être utilisées pour les images :

| Propriété | Description | Valeurs possibles |
|---|---|---|
| width | Dimension de l'image par rapport à elle-même | Pourcentage, valeurs fixes |
| width_space | Quantité d'espace horizontal que l'image doit couvrir, en pourcentage. `100%` signifie que l'image doit couvrir toute la largeur de la page même les marges. | Pourcentage |
| TODO | | |

## Les paragraphes-codes (pfb-code)

Ces paragraphes sont des paragraphes simples, contenant un seul "mot-programme", et permettent notamment de gérer le contenu du livre. Ce ne sont donc pas à proprement parler des paragraphes de texte mais ils auront une influence réelle sur le livre produit. On trouve par exemple :

```
Pour passer la suite à la page suivante :

(( new_page ))
```

```
Pour l'inscription de l'index :

(( index ))

Pour l'inscription de la table des matières :

(( tdm ))

Pour l'inscription d'une bibliographie :

(( biblio(films) ))

Etc.
```

---

## Numérotation des paragraphes

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | **:numerotation:** | `pages` (défaut), `parags` |
| | | **:num_parag:** | Table de valeurs |

Pour un livre technique, où les références sont fréquentes, ou si l'on veut que l'index ou les bibliographies renvoient à des endroits très précis du livre, il peut être intéressant de numéroter les paragraphes. Pour ce faire, on met la propriété `:parags` de la recette du livre ou de la collection à `true`.

```
book_format:
  text:
    numerotation: pages # ou parags
```

L'affichage utilise par défaut la police `Bangla`, mais elle peut être définie grâce à la propriété **`:num_parag`** de la recette, après s'être assuré que cette fonte était définie dans les fontes du livre ou de la collection :

{À refaire}

Le chiffre peut ne pas être tout à fait ajusté au paragraphe. Dans ce cas, on utilise la propriété `:top_adjustment` pour l'aligner parfaitement. La valeur doit être donnée en *pixels PDF*, elle doit être assez faible (attention de ne pas décaler tous les numéros vers un paragraphe suivant ou précédent.

```
:num_parag:
  # ...
  :top_adjustment: 1
```

Noter qu'on peut également demander à ce que la numérotation des pages se fasse sur la base des paragraphes et non pas des pages (pour une recherche encore plus rapide).

---

## Saut de page

***Prawn-for-book*** gère automatiquement les passages à la page suivante lorsque le texte arrive en bas de page. On peut cependant tout à fait forcer un saut de page pour forcer le passage à la page suivante à l'endroit voulu. On utilise dans le texte, ***seul sur un paragraphe***, l"une de ces deux marques :

```
(( new_page ))

<!-- OU -->

(( nouvelle_page ))
```

> Notez la forme d'une *commande Prawn-for-book* (elles permettent d'affiner l'impression du livre jusque dans le moindre détail) :
>
> - la double parenthèse
> - l'espace laissée de chaque côté de cette parenthèse, entre la commande et la parenthèse intérieure.

Si l'on veut se retrouver **sur une page paire**, utiliser l'une de ces marques :

```
(( new_even_page ))

ou

(( nouvelle_page_paire ))
```

Si l'on veut se retrouver sur une page impaire, utiliser l'une de ces marques :

```
(( new_odd_page ))

ou

(( nouvelle_page_impaire ))

ou

(( new_belle_page ))
```

---

## Insertion d'un texte externe

On peut insérer un autre fichier `pfb.md` (ou autre…) dans le texte `texte.pfb.md` d'un livre Prawn. Pour ce faire, il suffit d'utiliser la commande **`include:`** suivie du chemin relatif ou absolu du fichier.

Par exemple, si le dossier du livre contient un dossier `textes` et un fichier texte `introduction.pfb.md` contenant le texte de l'introduction, on peut l'insérer dans le livre à l'endroit voulu à l'aide de :

```
(( include: textes/introduction ))
```

Noter que ci-dessus aucune extension de fichier n'a été nécessaire. Elle n'est utile que s'il existe plusieurs fichiers de même affixe (nom sans l'extension) dans le dossier. Dans le cas contraire, **Prawn-for-book** recherche le fichier dont il est question.

---

## Headers & Footers (entêtes et pieds de page)

Par défaut (c'est-à-dire sans aucune précision), seul le pied de page est construit, avec le numéro de la page au milieu. Mais il est possible de définir finement chaque entête (*header*) et chaque pied de page (*footer*) et même d'en créer autant que l'on veut, tout à fait différents, pour les différentes sections du livre.

### Principe

Chaque **pied de page** (*footer*) et chaque **entête** (*header*) est une partie contenant trois sections appelées des "**TIERS**", un à gauche, un à droite et un au milieu de chaque page gauche et droite, où sont définis les éléments à afficher.

Pour gérer les entêtes et pieds de page, on crée des DISPOSITIONS qui comprennent les données suivantes :

- un nom humain pour mémoire,
- un rang de pages sur lequel appliquer la disposition,
- un *headfooter* pour l'entête des pages gauche et droite (cf. ci-dessous),
- un *headfooter* pour le pied des pages gauche et droite,
- une valeur d'ajustement vertical du pied de page et de l'entête,
- un identifiant.

On crée autant de dispositions que nécessaire.

### Rangs de pages

Une disposition est définie pour un rang de pages qui peut être défini explicitement grâce aux paramètres `:first_page` et `:last_page`.

## Contenu

Le contenu de chaque *TIERS*, quelconque, peut être :

- le numéro de la page,
- le nom du titre courant, de niveau 1, 2 ou 3
- un contenu textuel explicite (et invariable de page en page — par exemple la date de fabrication du livre-esquisse) — note : il peut contenir des variables ou du code à évaluer,
- une procédure évaluée à la levée

## Définition des entêtes et pieds de page

Pour définir les entêtes et les pieds de page, le mieux est d'utiliser l'assistant, c'est le meilleur moyen de ne pas faire d'erreur pour cette donnée sensible et complexe.

Pour lancer l'assistant, jouer `$> pfb assistant` et choisir "Assistant Header Footer".

## Positionnement

Pour bien régler la position des headers et footers, il faut comprendre qu'ils s'inscrivent toujours par rapport à la marge définie, dans cette marge (l"idée est que la marge définit donc toujours la vraie surface contenu du texte, que rien ne vient la rogner — sauf les numéros de paragraphes lorsqu"ils sont utilisés).

Pour les entêtes, ils sont inscrits 5 PS-Points au-dessus de la marge haute. Il faut donc que cette marge haute fasse au moins `5 + <hauteur de ligne d'entête>` (rappel : Prawn laisse toujours 10 PS-Points de fond perdu autour des pages).

**Affiner le positionnement** on joue sur la propriété `header_vadjust` et la propriété `footer_vadjust` de la disposition (qui se règle en ps-point). De cette manière, en jouant sur les marges hautes et basses et sur cette valeur, on peut avoir le positionnement exact désiré.

> Note : la valeur, avec l'assistant, peut aller de -20 à 20. Si on doit utiliser une autre valeur (ce qui n'est pas conseillé…) éditer la recette à la main.

## Tiers et contenus

Comme nous l"avons dit, on considère qu'un entête et un pied de page est divisé en deux fois trois "TIERS" occupant chacun un tiers de la largeur de la page, d'où leur nom. Pour définir un "headfooter", ces trois cases n'ont pas à être définis.

Ces tiers sont repérés par des clés qui portent en préfix l'indication de la page `pg_` pour "page gauche" et `pd_` pour "page droite" et en suffixe la position du tiers dans la page : `_left` pour le tiers à gauche, `_center` pour le tiers au center et `_right` pour le tiers à droite. On a donc :

```
---
:headers_footers:
  :headfooters:
    :HF0001:
      :id: :HF0001
      :name: Le headfooter en démo
      :font: Times-Roman
```

```
      :size: 12
      :style: null
      :pg_left:
        # ... définition... (il ne faut définir que les tiers utiles)
      :pg_center:
        # ... définition...
      :pg_right:
        :content: :titre1 # requise
        :align: :right
        :size: 40
        :font: Geneva
        :casse: :min # ou :all_caps, :keep, :title
      :pd_left:
        # ... définition...
      :pd_center:
        # ... définition...
      :pd_right:
        # ... définition...
```

### Dans la recette

```
---
# ...
#<headers_footers>
:headers_footers:
  :dispositions:
    # ... définition des dispositions (table)
  :headfooters:
    # ... défintion des headfooters (table
```

### Variables

On peut utiliser des variables à l'aide de `#{nom_de_la_variable}` dans un texte personnalisé (`:custom_text`).

---

## Pages spéciales

### Page de titre

La *page de titre* n'est pas à confondre avec la couverture (qui fait l'objet d'un fichier séparé pour un traitement différemment comme c'est souvent le cas). Il s'agit ici de la page, souvent après la page de faux titre et la page de garde qui présente toutes les informations générales sur le livre, titre, sous-titre, auteur, éditeur.

Pour sa mise en page, voir la recette concernant les pages spéciales.

## Page d'informations

Nous appelons "page d'informations" la page de fin de livre où sont présentés toutes les informations sur la conception du livre, metteur en page, correcteurs, imprimeurs, isbn et autre date de dépôt légal.

Pour définir les informations, ouvrir une fenêtre de Terminal au dossier du livre ou de la collection et utiliser l'assistant en jouant la commande `$> pfb assistant` et choisir "Assistant Page Infos".

Ces informations peuvent être réparties de 3 façons différentes :

- distribuées sur la page (réparties de façon égale sur la surface d'une page entière)
- en haut de page (toutes les informations sont rassemblées de façon compacte au-dessus d'une des dernières pages),
- en bas de page (toutes les informations sont rassemblées de façon compacte en bas d"une des dernières pages).

---

## Table des matières

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | :table_of_contents | Table de valeurs cf. |

La table des matières se construit sur la base des titres.

Elle s'inscrit dans le livre à l'endroit où est placé dans le texte un :

```
(( toc ))

<!-- OU -->

(( tdm ))
```

> "toc" signifie "Table of Contents" ou "Table des matières" en anglais.

**ATTENTION** : La construction de la table des matières n'ajoute pas automatiquement de nouvelles pages si la table déborde de la page qui lui est réservée (**FAUX***) (tout simplement parce qu"alors tous les numéros de pages seraient obsolètes…). Si la table des matières tient sur plusieurs pages, il faut donc ajouter autant de marques de nouvelles pages que voulus.

> *En fait, elle le fait maintenant, mais si la pagination après, ça n'est pas trop grave ? Non, en fait, il faudrait que soit calculé dans le premier tour le nombre de page pour la table des matières et qu'elle soit inscrite ensuite. Noter que si la tdm est inscrite à la fin du livre, il n'y a plus de problème.

Voir la partie Tous les types de pages qui définit la recette du livre.

Voir ici pour exclure un titre de la table des matières.

---

## Page d'index

Le plus simple pour construire un index dans un livre est d'utiliser la mise en forme par défaut, autant dans l'identification des mots à indexer que dans l'aspect de l'index final. Si l'on respecte ça, pour ajouter l'index, on a juste à insérer le texte suivant dans le texte du livre :

```
(( index ))
```

À l'endroit de cette marque sera inséré un index contenant tous les mots indexés dans le texte.

Par défaut, on repère les mots à indexer dans le texte par :

```
Ceci est un index:mot unique à indexer.

Ceux-là sont dans un index(groupe de mots) qu'il faut entièrement indexer.

Ce index(mot|verbe) doit être indexé avec le mot "verbe" tandis que :

Ces index(mots-là|idiome) doivent être indexé avec le mot "idiome".

# La barre "|" sert souvent pour séparer les données dans P4B.
```

Si l'on veut utiliser une autre méthode pour indexer les mots, on peut définir la méthode `__paragraph_parser(paragraph` du fichier `parser.rb` du livre ou de la collection.

cf. Parsing personnalisé du texte pour savoir comment parser les paragraphes pour en tirer les informations importantes.

Il s'agit donc, ici, de programmer la méthode `__paragraph_parser` pour qu'elle récupère les mots à indexer. Par exemple, si ces mots sont repérés par la balise `index:mot` ou `index:(groupe de mot)`, il suffit de faire :

```ruby
def __paragraph_parser(paragraph)

  # Note : @table_index a déjà été initiée avant
  paragraph.text.scan(/index[:\(](.+?)\)?/).each do |idiom|
    @table_index.key?(idiom[0]) || @table_index.merge!(idiom[0] => [])
    @table_index[idiom[0]] << {text: idiom, parag: paragraph}
  end
end
```

À l'issue du traitement, la table `@table_index` (de l'instance `PdfBook`) contiendra en clé tous les mots trouvés et en valeur une liste de toutes les itérations. Cette liste contiendra la liste des pages ou la liste des paragraphes en fonction du type de pagination adopté pour le livre ou la collection.

On peut donc faire ensuite :

```
module Prawn4book
  class PdfBook
    attr_reader :table_index
  end
end

module ParserParagraphModule
  def __paragraph_parser(paragraph)
    #... cf ci-dessus
  end
end

module PrawnCustomBuilderModule
  def __custom_builder(pdfbook, pdf)
    pdfbook.table_index.each do |idiom, occurrences|
      pdf.text "Index de '#{idiom}'"
      pdf.text occurrences.map {|oc| oc[:parag].numero }.uniq.join(', ')
    end
  end
end
```

## Pages de bibliographie

Voir la partie Tous les types de pages qui définit la recette du livre pour avoir un aperçu rapide des la définition d'une bibliographie.

> On peut obtenir un assistant à la définition des bibliographies du livre ou de la collection en jouant la commande :
>
> ```
> $> pfb aide biblio
> ```

Une bibliographie nécessite :

- de définir **la balise** qui va repérer les éléments dans le texte (par exemple `film` ou `livre`)
- de définir **un titre** qui sera utilisé dans le livre (`:title`),
- de définir le **chemin d'accès** à ses données (`:path`),
- de définir **la page** sur laquelle sera écrite la bibliographie,
- de définir **les données** utilisées par la bibliographie et qu'elles soient valides,
- de définir **la mise en forme** utilisée pour le livre pour présenter les informations sur les éléments.

### La balise de la bibliographie

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | **:bibliographies:** | null/table |

La *balise* est le mot qui sera utilisé pour repérer dans le texte les éléments à ajouter à la bibliographie. Par exemple, pour une liste de films, on pourra utiliser `film` :

```
Je vous parle d'un film qui s'appelle film(idFilmTitatic|Le Titanic) et se déroule dans
un bateau.
```

Elle est définit dans la propriété `:tag` dans le livre de recette du livre ou de la collection :

```
# in recipe.yaml
# ...
:bibliographies:
  :book_identifiant: 'livre'
  :biblios:
    film:
      # ...
```

Dans le texte, elle doit définir en premier argument l'identifiant de l'élément concerné dans les données.

Cette balise permettra aussi de définir la bibliographie à inscrire dans le livre, sur la page voulue, avec la marque :

```
(( bibliographie(film) ))

ou

(( bibliography(film) ))

ou

(( biblio(film) ))
```

Pour plus de détail, cf. la page de la bibliographie

**Le titre de la bibliographie**

Ce titre est celui qui apparaitra sur la page de bibliographie du livre. Il doit être défini entièrement, par exemple "Liste des films cités" ou "Liste des livres utiles".

Il est défini par la propriété `:title` dans la recette du livre ou de la collection.

```
# in recipe.yaml
# ...
:bibliographies:
  :biblios:
    film:
      :title: Liste des films cités
```

Par défaut, ce titre sera d'un niveau 1, c'est-à-dire d'un niveau grand titre. Mais on peut définir son niveau propre à l'aide de `:title_level:` :

```
# in recipe.yaml
# ...
:bibliographies:
  :biblios:
    film:
      :title: Liste des films cités
      :title_level: 3
```

**Chemin d'accès aux données de la bibliographie**

L'autre donnée absolument requise pour qu'une bibiographie soit opérationnelle concerne son `:path`, c'est-à-dire le chemin d"accès à ses données, donc le dossier contenant les fiches de ses items.

```
# in recipe.yaml
# ...
:bibliographies:
  :biblio:
    mabib:
      :title: Le Titre de MaBib
      :path: ./path/to/cards/folder
```

Comme on peut le voir, ce chemin peut être défini de façon relative (par rapport au dossier du livre, ou de façon absolue (ce qui n'est pas recommandé, si le dossier change de place plus tard ou si le dossier du livre est transmis..

**La page de la bibliographie**

On utilisera simplement la marque suivante pour inscrire une bibliographie sur la page :

```
(( biblio(<tag>) ))

ou (( bibliographie(<tag>) ))

ou (( bibliography(<tag>) ))
```

... où `<tag>` est la balise définie dans la recette du livre (propriété `:tag`.

Une bibliographie ne s'inscrit pas nécessairement sur une nouvelle page. Si ça doit être le cas, il faut placer le code `(( new_page ))` avant.

```yaml
# in recipe.yaml
# ...
:bibliographies:
  :biblios:
    film:
      :title: Liste des films
      :title_level: 2
```

> Noter que si le niveau de titre est 1 (ou non défini), et que les propriétés des titres de la recette définissent qu'il faut passer à une nouvelle page pour un grand titre, la bibliographie commencera alors automatiquement sur une nouvelle page.

**Les données de la bibliographie**

Les données bibliographiques sont contenus dans un dossier, par fiche (une fiche par item bibliographique) au format `yaml` ou `json`.

La source des données (le dossier) est indiquée dans le fichier recette du livre ou de la collection par la propriété `:path` :

```yaml
# in recipe.yaml
# ...
:bibliographies:
  :biblios:
    film: # le tag singulier
      :title: Liste des films
      :title_level: 2
      :path:  data/films
      :font: Fonte  # la fonte à utiliser
      :size: 10     # la taille de fonte (10 par défaut)
      :style: null  # éventuellement le style de la fonte
```

Ci-dessus, la source est indiquée de façon relative, par rapport au dossier du livre ou de la collection, mais elle peut être aussi indiquée de façon absolue si elle se trouve à un autre endroit (ce qui serait déconseillé en cas de déplacement des dossiers).

Pour le moment, *Prawn-for-book* ne gère que les données au format `YAML` et `JSON`. Ces données doivent produire une table où l'on trouvera en clé l'identifiant de l'élément et en valeur ses propriétés, qui seront utilisées pour la bibliographie. Par exemple, pour un fichier `films.yaml` qui contiendrait les données des films :

```yaml
# in data/films/titanic.yaml
---
titanic:
  title: The Titanic
  title_fr: Le Titanic
  annee: 1999
  realisateur: James CAMERON
```

```yaml
# in data/films/ditd.yaml
---
ditd:
  title: Dancer in The Dark
  annee: 2000
  realisateur: Lars VON TRIER

# etc.
```

**NOTE IMPORTANTE** : toute donnée bibliographique doit avoir une propriété `:title` qui sera écrite dans le texte à la place de la balise.

Voir ensuite dans la partie mise en forme la façon d'utiliser ces données.

**Mise en forme des données bibliographiques**

La mise en forme des bibliographies (ou de *la* bibliographie) doit être définie dans le fichier `formater.rb`.

Il faut y définir une méthode préfixée `biblio_` suivi par la balise (`:tag`) de la bibliographie concernée. Ce sera par exemple la méthode `biblio_film` pour la liste des films.

```ruby
# in formater.rb
module FormaterBibliographiesModule # attention au pluriel

  # Méthode mettant en forme les données à faire apparaitre et renvoyant
  # le string correspondant.
  def biblio_film(element) # l'element, ici, est un film, son instance
    c = []
    element.instance_eval do
      c << title
      c << " (#{title_fr})" if title_fr
      c << annee
    end
    return c.join(', ')
  end

  # Autre tournure possible
  def biblio_autre(element)
    '%{title.upcase} de %{writers}, %{year}' % element.data
  end

end #/module FormaterBibliographiesModule
```

Noter qu'avec cette formule, les données sont toujours présentées sur une ligne. À l'avenir, on pourra imaginer une méthode qui reçoit `pdf` (l'instance `{Prawn::View}`) et permette d'imprimer les données exactement comme on veut, même dans un affichage complexe.

Noter également qu'on n'indique pas, ici, les pages/paragraphes où sont cités les éléments, cette information est ajoutée automatiquement par l'application, après le titre et deux points. L'indication par page ou par paragraphe dépend du type de pagination adoptée dans le livre. En conclusion, le listing final ressemblera à :

```
<partie définie par biblio_tag> : <liste des pages/paragraphes séparés par des
virgules>.
<partie définie par biblio_tag> : <liste des pages/paragraphes séparés par des
virgules>.
<partie définie par biblio_tag> : <liste des pages/paragraphes séparés par des
virgules>.
```

---

## Références (et références croisées)

On peut faire très simplement des références dans le livre (références à d'autres pages ou d'autres paragraphes, du livre ou d'autres livres) à l'aide des balises :

```
(( <-(id_reference_unique) )) # référence (cible)

(( ->(id_reference_unique) )) # appel de référence
```

La référence sera tout simplement supprimée du texte (attention de ne pas laisser d'espaces — même si, normalement, ils sont supprimés). Pour l'appel de référence il sera toujours remplacé par *"la page xxx"* ou *"le paragraphe xxx"* en fonction de la pagination souhaitée et du préfix de référence choisi (TODO).

### Références croisées

Pour une *référence croisée*, c'est-à-dire la référence à un autre livre, il faut ajouter un identifiant devant la référence et préciser le sens de cet identifiant.

```
Pour trouver la référence croisée, rendez-vous sur la (( ->
(IDLIVRE:id_reference_unique) )).
```

Pour traiter une référence croisée, on a besoin de plusieurs choses :

- connaitre le livre en tant qu'entité bibliographique qui contiendra notamment les données qui seront ajoutées à la bibliographie (titre, auteurs, année, ISBN, etc.)
- connaitre le livre en tant que livre "Prawn-for-book", qui définira, dans son dossier, un fichier `references.yaml` contenant les références relevées lors de la dernière compilation du livre.
- connaitre la relation entre ces deux éléments (l'entité bibliographique et le livre pfb). Question : cette relation ne pourrait-elle pas être définie dans l'entité bibliographique ? ce qui permettrait de n'avoir qu'à définir cet entité, sans avoir à définir les deux derniers éléments.

Ces deux choses sont définies à un seul endroit : la fiche bibliographique du film ciblé. Cette fiche, en plus de `:title`, doit définir `refs_path` qui contient soit le chemin complet au fichier `references.yaml` des références, soit au dossier du livre, qui contiendra ce fichier lorsque le livre aura été construit.

#### Référence croisée vers un livre non prawn

On peut tout à fait faire référence à un endroit précis d'un livre quelconque non fabriqué par Prawn. Pour cela, il suffit de définir son fichier de référence "à la main", conformément à son format ci-dessous.

> Noter qu'il peut être difficile de connaitre le numéro de paragraphe dans un livre imprimé. Dans ce cas, laisser la donnée vide et, si les références se font par paragraphe, c'est exceptionnellement la donnée page qui sera utilisée).

Ce fichier peut être placé dans le dossier du livre lui-même, dans un dossier "livres_imprimes_pour_references", par exemple, et créer dedans des dossiers, au titre des livres, et dans ces dossiers, le fichier `references.yaml`.

### Fichier de références

Les références du livre sont enregistrées dans un fichier `references.yaml` qui permettra à d'autres livres d'y faire… référence.

Il est constitué de cette manière :

```
---
<cible_id>:
  page: <num page>
  paragraph: <num paragraph>
<cible_id>:
  page: <num page>
  paragraph: <num paragraph>
# etc.
```

---

## Exclure des paragraphes

Cf.ci-dessous les commentaires.

## Commentaires

Pour ajouter des commentaires dans un fichier texte destiné à l'impression, on le place entre commentaire markdown normaux.

```
<!-- Commentaire sur une ligne -->

<!--
Commentaires
Sur plusieurs
Lignes
-->
```

Il est donc tout à fait possible d'exclure du texte en le mettant entre ces signes :

```
# Titre principal
<!--
## Titre zappé
Paragraphe zappé, non imprimé
-->
## Un titre pris en compte.
```

Noter que par rapport à du markdown pur, il est inutile de laisser des lignes vierges entre les types de paragraphes.

---

## Méthodes de traitement et de formatage propres

*Prawn-for-book* utilise 3 moyens de travailler avec les paragraphes au niveau du code :

- un module de formatage personnalisé (`formater.rb`),
- un module de méthodes d'*helpers* qui permettent un traitement ruby personnalisé (`helpers.rb`),
- un module de méthode de `parsing` qui traite de façon propre le paragraphe (`parser.rb`).

Ces trois fichiers (`parser.rb`, `helpers.rb` et `formater.rb`) sont propres à chaque livre ou chaque collection et seront toujours automatiquement chargés s'ils existent.

### Méthode d'helpers — `(( #<method>(<args>) ))`

Les méthodes d'helpers s'utilisent dans le texte comme un code ruby :

```
Ceci est un texte de paragraphe avec un (( #code_ruby_simple )) qui sera évalué.

Ceci est un paragraphe avec qui devra apprendre à dire (( #code_ruby("bonjour tout le monde") )).
```

> Attention : ne pas oublier les espaces à l'intérieur des parenthèses, comme c'est le cas avec le signe de Prawn, les doubles parenthèses.

Cette méthode ou variable `code_ruby_simple` doit être définie en *Ruby* dans le fichier `helpers.rb` du [livre][] ou de la [collection][] de la manière suivante :

```ruby
# in ./dossier/livre/helpers.rb
module PrawnHelpersMethods
  def code_ruby_simple
    # On utilise ici 'pdfbook' et 'pdf' pour obtenir le livre ou
    # son builder.
    # On retourne la position actuelle du curseur dans le fichier
    # pdf en l'arrondissant :
    return round(pdf.cursor)
  end

  def code_ruby(str)
    return "« #{str} »"
```

```
  end
end
```

Ces méthodes d'helpers doivent obligatoirement retourner le code (le texte) qui sera écrit à leur place dans le paragraphe.

Les seules conventions a respecter ici sont :

- le fichier doit impérativement s'appeler `helpers.rb` (au pluriel, car il y a plusieurs *helpers* mais l'application cherchera aussi le singulier),
- le fichier doit impérativement se trouver à la racine du dossier du livre ou du dossier de la collection (les deux seront chargés s'ils existent — attention aux collisions de noms),
- le titre du module doit être **`PrawnHelpersMethods`** (noter les deux au pluriel et là c'est impératif).

Les méthodes ont accès à **`pdfbook`** et **`pdf`** qui renvoient respectivement aux instances `Prawn4book::PdfBook` et `Prawn4book::PrawnView`. La première gère le livre en tant que livre (pour obtenir son titre, ses auteurs, etc.) et la seconde est une instance de `Prawn::View` (substitut de `Prawn::Document`) qui génère le document PDF pour l'impression.

On peut par exemple obtenir le numéro de la page avec `pdf.page_number` et la consigner :

```
Ceci est un paragraphe avec au bout un code qui sera caché (remplacé par un string
vide) pour savoir le numéro de cette page et le numéro de ce paragraphe.((
#consigne_page('page_a_memoriser') ))(( #consigne_paragraphe('par2memo') ))
```

… avec les deux méthodes d'helpers définies ainsi :

```
# in helpers.rb
module PrawnHelpersMethods
  def consigne_page(id)
    @pages_memorisees ||= {}
    @pages_memorisees.merge!(id => pdf.page_number)
    return ''
  end

  def consigne_paragraphe(id)
    @paragraphes_memorises ||= {}
    @paragraphes_memorises.merge!(id => pdf.paragraph_number)
    return ''
  end
end
```

Grâce à `pdfbook`, on a accès à l'intégralité des valeurs de la recette. Ce qui signifie qu'on peut consigner n'importe quelle valeur dans la recette, qu'on pourra récupérer dans ces helpers. Par exemple, si on définit dans la recette :

```
# in recipe.yaml
---
# ...
:ma_couleur_preferee: '#2569F8'
:une_autre_couleur:   '#45DF56'
```

... alors on pourra utiliser dans le helper :

```
module PrawnHelpersMethods
  # @return le texte +str+ en le mettant à la couleur +which_color+ qui est
  # une couleur hexa définie dans la recette du livre
  def colorise(str, which_color)
    code_couleur = pdfbook.recipe.get(which_color)
    return "<font color=\"#{code_couleur}\">#{str}</font>
  end
end
```

... et l'utiliser dans le texte avec :

```
Ce paragraphe contient un (( #colorise("texte", :ma_couleur_preferee) )) qui sera dans
ma couleur préférée et un (( colorise("autre texte", :une_autre_couelur) )) qui sera
dans une autre couleur.
```

Ce texte, une fois construit, produira :

TODO: montrer l'image produite.

## Formatage personnalisé (`formater.rb`)

### Formatage des paragraphes

Le principe est le suivant :

```
SI un paragraphe commence par une balise (un mot suivi sans espace par '::')
    par exemple : "custag:: Le texte du paragraphe."

ALORS ce paragraphe sera mis en forme à l'aide d'une méthode de nom :

    __formate_<nom balise>

    par exemple : def __formate_custag(string)

QUI SERA DÉFINIE dans le fichier 'formater.rb' définissant le module
'FormaterParagraphModule'
```

```ruby
# in ./formater.rb
module FormaterParagraphModule # Ce nom est absolument à respecter
  def __formate_custag(string)
    # ...
    return string_formated
  end
end

module FormaterBibliographiesModule # ce nom est absolument à respect
end #/module
```

Ce code doit être placé dans un fichier **`formater.rb`** soit dans le dossier du livre soit dans le dossier de la collection si le livre appartient à une collection.

> Noter que si collection et livre contient ce fichier, seul celui de la collection sera chargé.

**Formatage des éléments de bibliographie**

Le formatage est défini dans des méthodes `biblio_<tag>` dans un module **`FormaterBibliographiesModule`** du fichier `formater.rb`:

```ruby
# in formater.rb

module FormaterBibliographiesModule
  def biblio_film(film)
    # ...
  end
end
```

Cf. la section “mise en forme de la bibliographie” pour le détail.

---

## Parsing personnalisé des paragraphes (`parser.rb`)

De la même manière que les paragraphes sont formatés (cf. ci-dessus), ils peuvent être parsés pour en tirer des informations utiles (pour faire un index, une bibliographie, etc.)

Il suffit pour cela de créer un fichier de nom `parser.rb` dans le dossier du livre (ou de la collection) qui contienne :

```ruby
module ParserParagraphModule # ce nom est absolument à respecter
  def __paragraph_parser(paragraphe)
    # Parse le paragraphe {PdfBook::NTextParagraph}
    str = paragraphe.text
  end
  # ...
end #/module

module PrawnCustomBuilderModule # ce nom est absolument à respecter
```

```
  #
  # Ici doit être défini les choses à faire avec les informations
  # qui ont été parsées
  #
  def __custom_builder(pdfbook, pdf)
    #
    # P.e. pour insérer une nouvelle page avec du texte
    #
    pdf.start_new_page
    pdf.text "Ceci est un texte avec les infos parsées."

  end
end #/module
```

> Pour réaliser le texte des nouvelles pages, cf. blocs de texte avec Prawn.

Ce fichier contient donc deux modules :

- **ParserParagraphModule** définit la méthode `__paragraph_parser` qui parse les paragraphes.
- **PrawnCustomBuilderModule** définit la méthode `__custom_builder` qui construit les éléments du livre en rapport avec les informations relevées.

---

# Recette du livre ou de la collection

## Définition

La *recette du livre* permet de définir tous les aspects que devra prendre le livre, c'est-à-dire le fichier PDF prêt-à-imprimé. On définit dans ce fichier les polices utilisées (à empaqueter), les marges et la taille du papier, les titres, les lignes de base, le titre, les auteurs, etc.

## Création de la recette du livre

On peut créer de façon assistée la recette d'un livre en ouvrant un Terminal dans le dossier où doit être initié le livre — ou le dossier où se trouve déjà le texte, appelé `texte.pfb.txt` ou `text.pfb.md` — et en jouant la commande : `> prawn-for-book init`.

Cette commande permet de créer un fichier `recipe.yaml` contenant la recette du livre ou de se servir d'un modèle prérempli. Passons en revue les différentes paramètres à régler.

## Contenu de la recette du livre

### Informations générales

```
:book_title:  Le titre du livre
:book_subtitle: |
  Sous-titre qui sera ajouté sous le titre dans la
  page de titre. Tous les retours chariots ici
  seront reproduits tels quels.
:collection:  false # true => appartient à une collection
:auteurs:     ['Prénom NOM', 'Prénom NOM']
:book_id:     identifiant_simple # nom du dossier par exemple
:main_folder: "/path/to/folder/principal/du/livre"
:text_path:   true  # pour dire texte.pfb.txt ou texte.pfb.md dans le
                    # dossier du livre, sinon le path absolu
```

## Informations générales pour une collection

```
:name: "Nom humain de la collection"
:short_name: "Nom raccourci" # pour les messages seulement
:main_folder: "/path/to/folder/principal/de/la/collection"
```

## Aspect général du livre

```
:dimentions: ['210mm', '297mm'] # Ne pas oublier les unités
:layout: :portrait # ou :landscape
:marges:
  :top: '20mm'  # marge haut
  :bot: '20mm'  # marge bas
  :ext: '30mm'  # marge extérieure (*)
  :int: '15mm'  # marge intérieure (*)
:background: "/path/to/image/fond.jpg" # pour une image de fond
```

> (*) Prawn4Book est spécialement désigné pour créer des livres papier, donc les marges sont toujours définies avec la marge intérieure (côté pliure) et la marge extérieure (côté tranche — le vrai sens de "tranche").

## Aspect des pages

```
:default_font:      FontName  # font par défaut
:default_font_size: 11        # taille de fonte par défaut
:default_style:     :normal   # style par défaut
:line_height:       12.5      # Hauteur de la ligne de référence
:leading:           1         # Espacement par défaut entre les mots
:num_page_style:    num_page  # Type de numérotation (
                              # 'num_page'  => par numéro de page
                              # 'num_parag' => par numéro de paragraphes
:opt_num_parag:     false     # si true, on numérote les paragraphes.
```

La donnée `:line_height` est particulièrement importante puisqu'elle détermine où seront placées toutes les lignes du texte dans le livre, sauf exception [[AJOUTER RENVOI VERS CETTE EXCEPTION]]. Elle permet de définir la **grille de références** c'est-à-dire la grille sur laquelle seront alignées toutes les lignes de texte pour produire un livre professionnel.

## Données des titres

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | **:titles:** | Table par titre |

```
:titles:
  :level1:
    :next_page: true    # true => nouvelle page pour ce titre
    :belle_page: false  # mettre à true pour que le titre soit
                        # toujours sur une belle page (impaire)
    :font: "LaFonte"
    :size: 30
    :lines_before: 0
    :lines_after: 4
    :leading: -2
  :level2:
    # idem
  :level3:
    # idem
  # etc.
```

Les `lines_before` et `lines_after` se comptent toujours en nombre de lignes de référence, car les titres sont toujours alignés par défaut avec ces lignes (pour un meilleur aspect). On peut cependant mettre une valeur flottante (par exemple `2.5`) pour changer ce comportement et placer le titre entre deux lignes de référence.

- par défaut, les titres se placent toujours sur des lignes de référence,
- on parle toujours du **nombre de lignes ENTRE les éléments**. C'est-à-dire que si `:lines_after` est réglé à 4 pour un titre, on trouvera 4 lignes entre ce titre et l'élément suivant. Donc l'élément suivant sera posé sur une 5e ligne
- le `:line_before` d'un titre suivant s'annule si le titre précédent en possède déjà un. Si par exemple le titre de niveau 2 possède un `:lines_after` de 4 et que le titre de niveau 3 possède un `:lines_before` de 3, alors les deux valeurs ne s'additionnent pas, la première (le `:lines_after` du titre de niveau 2) annule la seconde (le `:lines_before` du titre de niveau 3).

  > Bien noter que c'est vrai dans tous les cas. Par exemple, si un titre de niveau 1 a son `:lines_after` réglé à 0, un titre de niveau supérieur aura beau avoir son `:lines_before` réglé à 4 ou 6, le titre de niveau supérieur sera "collé" au titre de niveau 1.

- on peut mettre une valeur flottante pour qu'un titre se place entre deux lignes de référence

La valeur du `leading` permet de resserrer les lignes du titre afin qu'il ait un aspect plus "compact", ce qui est meilleur pour un titre. Ne pas trop resserrer cependant.

## Données de l'éditeur/éditions

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | **:publisher:** | Table de données |

> Utiliser plutôt l'assistant : `$> pfb assistant` et choisir "publisher" ou "maison d'édition".

```
:publisher:
  :name:  "Nom édition" # p.e. "Icare Éditions"
  :adresse: |
    Numéro Rue de la voie
    XXXXX La Ville
  :site:    "https://site_editions.org"
  :logo:    "path/rel/or/abs/to/logo.svg"
  :siret:   null # Ou numéro de siret
  :mail:    mail@toedition.org
  :contact: contact@toedition.org
```

## Fontes

| | Recette | propriété | valeurs possibles |
|---|---|---|---|
| | | **:fonts:** | Table de données |

On peut être assister pour la création de la donnée des fontes (qui nécessite de connaitre les chemins d'accès à toutes les fontes possibles) de cette manière :

- ouvrir un Terminal au dossier du livre ou de la collection
- jouer la commande `$> pfb aide fontes` ou `$> pfb assistant` et choisir "Fontes".

```
:fonts:
  <nom utilisé>
    :<style>: "/path/to/font.tff"
    :<style>: "/path/to/font.tff"

# etc.
```

Par exemple :

```
# ...
```

```
# Une variable pour simplifier
dossier_fonts: &dosfonts "/Users/philippeperret/Library/Fonts"
fonts_system:  &sysfonts "/System/Library/Fonts"
prawn_fonts: &pfbfonts "/Users/philippeperret/Programmes/Prawn4book/resources/fonts"

# Définition des fontes (note : ce sont celles par défaut quand on
# utilise les templates)
#<fontes>
:fonts:
  Garamond:
    :normal: "*dosfonts/ITC - ITC Garamond Std Light Condensed.ttf"
    :italic: "*dosfonts/ITC - ITC Garamond Std Light Condensed Italic.ttf"
  Bangla:
    :normal: "*sysfonts/Supplemental/Bangla MN.ttc"
    :bold:   "*sysfonts/Supplemental/Bangla MN.ttc"
  Avenir:
    :normal: "*sysfonts/Avenir Next Condensed.ttc"
  Arial:
    :normal: "*dosfonts/Arial Narrow.ttf"
  Nunito:
    :normal: "*pfbfonts/Nunito_Sans/NunitoSans-Regular.ttf"
    :bold:   "*pfbfonts/Nunito_Sans/NunitoSans-Bold.ttf"
#</fontes>
```

> L'ordre des fonts ci-dessous peut être défini avec soin, car si certains éléments du livre ne définissent pas leur fonte, cette fonte sera choisie parmi les fontes ci-dessus. Pour des textes importants (comme les index, la table des matières, etc.) c'est la première fonte qui sera choisie tandis que pour des textes mineurs (numéros de paragraphes, entête et pied de page, etc.), c'est la seconde qui sera choisie.

---

### Tous les types de page

(c'est-à-dire la page à la fin du livre présentant les différentes informations sur ce livre)

{TODO À reprendre en totalité}

```
:skip_page_creation:  true  # à true, la première page automatique
                            # n'est pas générée (ce qui permet de
                            # contrôler cette première page)
:page_de_garde:       true
:page_de_titre:       true  # true => affichage de la page de titre
                            # avec les données par défaut
  # Sinon, on peut définir les données précisément
  # :sizes:  # Pour les tailles des différents éléments
  #   :collection_title: 14
  #   :title: 34
  #   :subtitle: 20
  #   :author: 16
```

```
  #   :publisher: 14
  # :spaces_before: # pour les "espaces avant" les éléments, en
  #                 # nombre de lignes de référence
  #   :title: 5 # en nombre de lignes de référence
  #   :subtitle: 1
  #   :author: 2
  # :logo:
  #   :height: 10 # hauteur du logo en millimètres
  #   # noter que le logo est toujours placé le plus en bas possible
  #   # On peut forcer un ':logo: false' pour forcer à ne pas afficher
  #   # le logo (s'il est défini dans la partie ':publisher')

:faux_titre:          true
  # On peut aussi définir la fonte et la taille :
  # :font:  "LaFonte"
  # :size:  16
# --- Réglage de la table des matières ---
# Elle sera affichée à la marque '(( toc ))'
:table_of_contents:
  :title:   "Table des matières" # si false, pas de titre
  :title_level:   2           # niveau de titre du titre ci-dessus
  :font:          'fontName'  # police à utiliser (elle doit être chargée)
  :size:          11          # taille de la police
  :line_height:   null        # Hauteur de ligne. Par défaut, c'est 14
  :first_line:    null        # Hauteur de la première ligne par rapport
                              # au bord haut de la page (hors marge)
  :add_to_numero_with: null   # largeur (en unité pdf) à ajouter au
                              # numéro de page. Correspond à l'arrêt
                              # des pointillés reliant les titres aux
                              # numéros des pages/paragraphes
  indent_per_offset: null     # ou liste des indentations en fonction
                              # du niveau de titre. Par défaut :
                              #   [0, 2, 4, 6, 8]
# --- Réglage de la page d'infos (fin du livre) ---
:infos:
  :display:     true      # pour la produire dans le livre
  :isbn:        null
  :depot_bnf:   3e trimestre 2022
  :cover:       "MM & PP" # auteurs de la couverture
  :mep:         ['Prénom NOM']  # mise en page
  :conception:  ['Prénom NOM']  # conception du livre
  :corrections: ['Prénom NOM']
  :print:       'Imprimé à la demande'

# --- Réglage des pages de bibliographie ---
:biblio:
  - :tag: livre       # tag utilisée dans le texte pour identifier les
                      # éléments à bibliographier
    :new_page: true   # true => mettre cette bibliographie sur une
```

```
                    # nouvelle page
  :title: "Liste des ?"  # titre utilisé sur la page
  :data:  "biblio/livres.yaml" # source des données
```

---

# Annexe

## Points PDF

Par défaut, les valeurs sont comprises en *points-PDF*. La valeur 12, par exemple, sera considérée comme “12 points-PDF”.

Mais on peut tout à fait utiliser d’autres mesures en ajoutant l”unité après la valeur, séparée par un point (**pas une espace**). Par exemple :

```
12.mm # pour 12 millimètre
1.3.cm # pour 1 centimètre et 3 millimètre
# etc.
```

Les unités possibles sont : `mm` (millimètres), `cm` (centimètres), `dm` (décimètres), `ft` (unités impériales — anglaises), `pt` (points).

# Ne pas afficher les espaces insécables

Pour ne pas afficher les espaces insécables dans Sublime Text :

- Sublime Text > Préférences > Settings - Syntax specific
- ajouter dans la fenêtre droite :

  ```
  {
      "draw_unicode_white_space": "none",
  }
  ```

- enregistrer.

# Package Sublime Text

Pour travailler le texte, le mieux est d’utiliser un éditeur de texte. Sublime Text est mon éditeur de choix et on peut trouver dans le dossier `./resources/Sublime Text/` un package `Prawn4Book` qu’on peut ajouter au dossier `Packages` de son éditeur (dans Sublime Text, activer le menu “Sublime Text > Préférences > Browse packages…” et mettre le dossier `Prawn4Book` dans le dossier `Packages`.

L’application reconnaitra alors automatiquement les fichiers `.pfb.txt` et utilisera un aspect agréable, tout mettant en exergue les éléments textuels particuliers (comme les balises de formatage des paragraphes).

## Choix d'une autre police

Plus tard, la procédure pourra être automatisée, mais pour le moment, pour modifier la police utilisée dans le document `.pfb.txt` (ou markdown), il faut éditer le fichier `Prawn4Book.sublime-settings` du package et choisir la `"font_face"` qui convient (en ajouter une si nécessaire). Régler aussi le `"font_size"` et `"line_padding_top"` pour obtenir le meilleur effet voulu pour un travail confortable sur le texte.

On peut ouvrir ce package dans Sublime Text à l'aide de :

```
$> pfb open package-st
```

.

# Prawn

## Blocs de texte avec Prawn

[fichier `helper.rb`]: