Prawn4book
Manuel

**Prawn4book
Manuel**

- Initiation d'un livre
- Ajouter un livre à une collection
- Initiation d'une nouvelle collection
- Construction du PDF du livre
  - Forcer le re-parsing du texte
  - Pour enregistrer le fichier `texte.yaml`
  - Ouvrir le fichier PDF produit
  - Position curseur des paragraphes
- Ouverture du PDF
- Texte du livre
  - Package Sublime Text
  - Modifier l'aspect visuel du texte dans Sublime Text
- Pages du livre
  - Les marges
  - Pages spéciales
    - Table des matières
    - Page d'index
    - Page de bibliographie
      - La balise de la bibliographie
      - Le titre de la bibliographie
      - La page de la bibliographie
      - Les données de la bibliographie
      - Mise en forme des données bibliographiques
- Contenu du livre (les paragraphes)
  - Définition
  - Les différents types de paragraphe
  - Exclure des paragraphes
  - Commentaires
  - Méthodes de traitement et de formatage propres
    - Méthode d'helpers
    - Formatage personnalisé (`formater.rb`)
      - Formatage des paragraphes
      - Formatage des éléments de bibliographie
    - Parsing personnalisé des paragraphes (`parser.rb`)
- Recette du livre
  - Définition
  - Création de la recette du livre
  - Contenu de la recette du livre
    - Informations générales
    - Aspect général du livre
    - Fontes

        Entête et pied de page
        Tous les types de page
    Disposition
    Variables
Annexe
Ne pas afficher les espaces insécables
Package Sublime Text
    Choix d'une autre police
Prawn
    Blocs de texte avec Prawn

Prawn4book est une application en ligne de commande permettant de transformer un texte en PDF prêt pour l'impression, grâce au (lovely) gem `Prawn`.

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

Ou son raccourci :

```
$> pfb
```

Tous les exemples de ce manuel présupposent qu'un alias de la commande a été créé, 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
```

> Noter ci-dessus que la commande sera `prawn-for-book` (qui est plus simple à taper)

Initiation d'un livre

- Créer un dossier dans lequel seront mis tous les éléments du livre,
- ouvrir une fenêtre Terminal dans ce dossier,
- jouer la commande `$> prawn-for-book init`,
- choisir de construire un nouveau livre,
- suivre le processus proposé et choisi.

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).

Initiation d'une nouvelle collection

- Créer le dossier dans lequel doit être placée la collection,
- ouvrir une fenêtre Terminal à ce dossier,
- jouer la commande `$> prawn-for-book init`,
- choisir de construire une 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
> prawn-for-book build
```

Certaines options permettent de travailler le livre avant sa fabrication définitive :

Pour s'arrêter à une page préciser, par exemple la 24e si on veut faire des essais minimum avec KDP :

```
$> prawn-for-book build -last=24
```

Forcer le re-parsing du texte

Par défaut, l'application utilisera le fichier `texte.yaml` s'il existe. Pour détruire ce texte, ce qui provoquera le reparsing du texte original, ajouter l'option `--force`.

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

Pour enregistrer le fichier `texte.yaml`

Si l'on est sûr de ne plus devoir parser le texte à chaque fois et qu'on veut préciser les choses au niveau de l'impression, on peut ajouter l'option `-save` (ou `--save`) à la commande `build` :

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

Cela provoquera une version *Prawn4book* du texte qui permettra de préciser le formatage.

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
```

Position curseur des paragraphes

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 :

```
$> prawn-for-book 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 `.p4b.txt` ou `.p4b.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
```

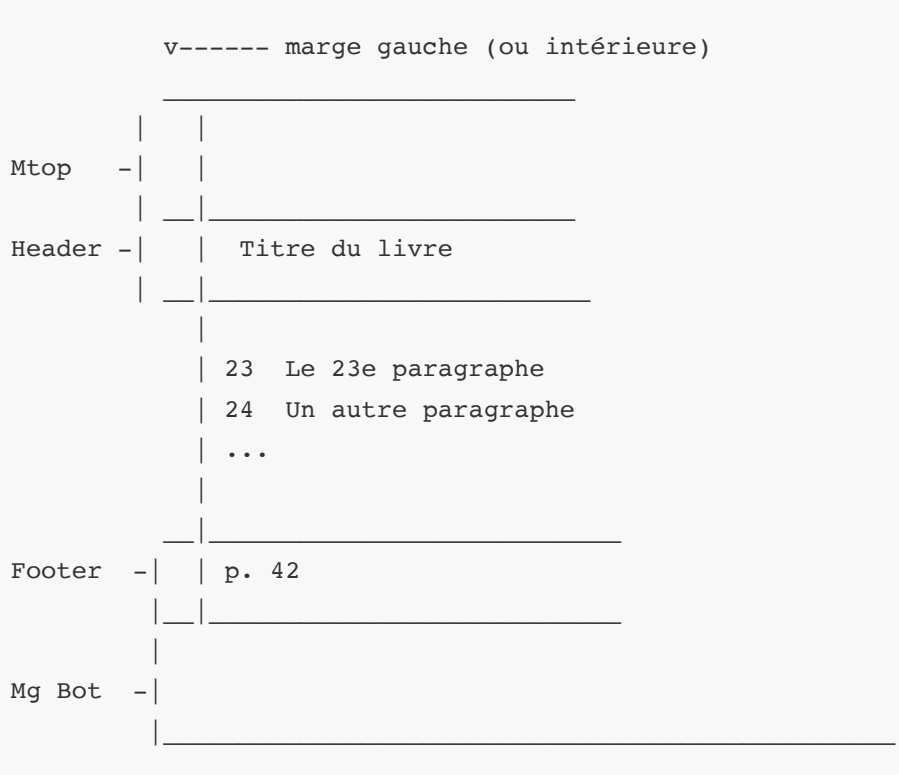
Modifier l'aspect visuel du texte dans Sublime Text

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).

Pages du livre

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.

Pages spéciales

Table des matières

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

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

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 :

```
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.

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
```

Page 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.

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,
- 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

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
# ...
:biblio:
  - :tag: 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 :

```
(( 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
# ...
:biblio:
  - :tag: 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
# ...
:biblio:
  - :tag: film
    :title: Liste des films cités
    :title_level: 3
```

La page de la bibliographie

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

```
(( biblio(<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 l'indiquer explicitement avec le réglage `new_page: true` dans la recette.

```
# in recipe.yaml
# ...
:biblio:
  - :tag: film
    :title: Liste des films
    :title_level: 2
    :new_page: true # => sur une nouvelle page
```

> 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

Il y existe deux moyens de définir les données d'une bibliographie :

- par fichier unique (l'extension indique comme les lire)
- par fiches séparées (dans un dossier)

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

```
# in recipe.yaml
# ...
:biblio:
  - :tag: film
    :title: Liste des films
    :title_level: 2
    :data:  data/films.yaml
```

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`. 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.yaml
---
titanic:
  title: The Titanic
  title_fr: Le Titanic
  annee: 1999
  realisateur: James Cameron
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

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.

Contenu du livre (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

```
Simple paragraphe

    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.

image

    Définit dans le texte par 'IMAGE[<data>]'
    Comme pour les simples paragraphes elle peut être précédée par des
    étiquettes de définition.

titre
    Définit dans le texte par '#[#[#]] Titre'
```

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

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

```
Ceci est un texte de paragraphe avec un #{code_ruby} qui sera évalué.
```

Cette méthode ou variable `code_ruby` doit être définie 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
    # 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
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 plusieur, 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),
- le titre du module doit être **`PrawnHelpersMethods`** (noter les deux au pluriel et là c'est impératif).

Chaque méthode peut utiliser `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.

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
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

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.p4b.txt` ou `text.p4be.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
: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.p4b.txt ou texte.p4b.md dans le
                    # dossier du livre, sinon le path absolu
```

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 (*)
```

> (*) 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").

```
:default_font:      FontName  # font par défaut
:default_font_size: 11        # taille de font par défaut
:line_height:       12.5      # Hauteur de la ligne de référence
:num_page_style:    num_page  # Type de numérotation (
                              # 'num_page'  => par numéro de page
                              # 'num_parag' => par numéro de paragraphes
```

Cette 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**.

Fontes

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 `$> prawn-for-book aide fontes`.

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

# etc.
```

Par exemple :

```
# ...
dossier_fonts: &dosfonts "/Users/philippeperret/Library/Fonts"
:fonts:
  Garamond:
    :normal: "*dosfonts/ITC - ITC Garamond Std Light Condensed.ttf"
    :italic: "/Users/philippeperret/Library/Fonts/ITC - ITC Garamond Std Light
Condensed Italic.ttf"
  Bangla:
    :normal: "/System/Library/Fonts/Supplemental/Bangla MN.ttc"
    :bold:   "/System/Library/Fonts/Supplemental/Bangla MN.ttc"
  Avenir:
    :normal: "/System/Library/Fonts/Avenir Next Condensed.ttc"
  Arial:
    :normal: "/Users/philippeperret/Library/Fonts/Arial Narrow.ttf"
```

Entête et pied de page

On peut définir les entêtes et les pieds de page dans le fichier recette du livre ou de la collection grâce aux données `:headers` et `:footers`.

> Noter qu'ils sont au pluriel

Le principe est que pour chaque rang de page on peut définir un pied de page et une entête différents.

```
:default: &styleheader
  :font: NomDeLaFont
  :size: 13.5
:headers:
  # Nom de l'entête, juste pour info, pour savoir ici ce que c'est
  - :name:    "Nom de ce premier rang"
    # Définition des pages qui utiliseront cet entête. Un rang de la
    # première page à la dernière.
    :pages: (12..15)
    # Disposition de l'entête. Il est toujours constitué de 3 sections,
    # le milieu, le côté gauche et le côté droit. Ils sont délimités
    # par des "|". Le tiret '-' permet de définir l'alignement dans cette
    # section :
    #   -mot    => alignement à droite
    #   mot-    => alignement à gauche
    #   -mot-   => alignement au centre
    :disposition:   '%titre1- | | -%titre2'
    # La police à utiliser. Elle doit impérativement être défini dans
    # les :fonts:
    :font:  Arial
    # Taille de la police (en points, je crois)
    :size:  11
  # Un autre rang
  - :name:    'Nom de ce second rang' # juste pour information
    # etc.
#
# --- PIEDS DE PAGE ---
#
# Les définitions sont les mêmes que pour les entêtes.
:footers:
  - :name: "Pied de page pour l'introduction"
    :pages: (1..5)
    :disposition:    ' | -%num- | '
    :font: Arial
    :size: 9
```

Tous les types de page

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

```
:page_de_garde:       true
:page_de_titre:       true
:faux_titre:          true
# --- Réglage de la table des matières ---
:table_des_matieres:
  :display: false # true pour l'afficher
  :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
  :from_top:    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
```

Disposition

Le pied de page et l'entête sont divisés en trois parties de taille variable en fonction du contenu. Dans le format (`:disposition`), ces trois parties sont définies par des `|`.

L'**alignement** s'indique par des tirets avant, après ou de chaque côté du contenu. Quand le tiret est à droite (`mot-`), le mot est aligné à gauche, quand le tiret est à gauche (`-mot`) le contenu est aligné à droite, quand les tirets encadrent le contenu (`-mot-`) — ou quand il n'y en a pas — le contenu est centré.

Variables

Les variables utilisables dans les entêtes et pieds de page sont toujours des mots simples commençant par `%`.

Pour les **niveaux de titre**, on utilise `%titre<NIVEAU>` par exemple `%titre4` pour les titres de niveau 4.

Pour les **numérotations**, on utilise `%num`. Noter que le contenu dépendra de la donnée `:num_page_style` de la recette du livre ou de la collection qui définit avec quoi il faut numéroter. TODO: à l'avenir on pourra imaginer avoir des numéros différents suivant les parties.

Annexe

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 `.p4b.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 `.p4b.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