

AsciiDocpro (v3.0.0)

Auteur : Emmanuel Ravrat

Livre



A s c i i d o c P r o

Table des matières

1. Lien entre AsciiDoc et AsciiDocpro	1
1.1. Qu'est-ce que AsciiDoc ?	1
1.2. Qu'est-ce qu'AsciiDocpro ?	1
1.3. Pourquoi adopter AsciiDocpro ?	1
1.4. La philosophie d'AsciiDocpro.	2
1.5. Comment mettre en place AsciiDocpro ?	3
1.6. Comment utiliser AsciiDocpro ?	3
2. AsciiDoc, AsciiDoctor, AsciiDocpro, AsciiDocproM, c'est quoi ?	4
3. Ecrire un chapitre	4
3.1. Où écrire un chapitre ?	5
3.2. Règles à respecter pour débiter un fichier de chapitre	5
3.3. Bien délimiter le contenu d'un chapitre	6
3.3.1. Rendre les chapitres atomiques	6
3.3.2. Organiser les répertoires d'un dossier de chapitre	7
3.3.3. Utiliser des images dans un chapitre	7
4. Ecrire un exercice	8
4.1. Où écrire un exercice ?	8
4.2. Règles à respecter pour débiter un fichier d'exercice	8
4.3. Organiser les répertoires d'un dossier d'exercice	9
4.4. Utiliser des images dans un exercice	10
5. Créer un "livre"	10
5.1. C'est quoi un livre dans AsciiDocpro ?	10
5.2. Où écrire un livre ?	10
5.3. Règles à respecter pour débiter un fichier de livre.	11
5.4. Les différents "livres" qu'il est possible de créer	11
5.4.1. La structure prise en exemple	12
5.4.2. Créer un livre avec plusieurs chapitres.	12
5.4.3. Un livre avec des chapitres et des exercices.	14
5.4.4. Un livre qui ne contient que des exercices.	15
5.4.5. Un livre avec un seul chapitre	16
5.4.6. Un livre avec un seul chapitre et ses exercices	16
5.5. Marquer un livre comme terminé	17
6. Personnaliser le rendu d'un document	18
6.1. Qu'est-il possible de personnaliser ?	18
6.2. Configurer les attributs d'application au niveau global	18
6.3. Configurer les attributs au niveau d'un "livre"	23
7. Utiliser ses propres attributs	24
7.1. Définir des attributs personnalisés au niveau global	24

7.2. Utiliser des attributs définis au niveau d'un livre	25
8. Utiliser un thème personnalisé	26
9. Les templates de code qu'il faut adopter	27
9.1. Qu'est-ce qu'un template de code ?	27
9.2. Template de code pour les questions et les réponses	27
9.2.1. Le code pour poser une question	27
9.3. Le code pour écrire une réponse	28
9.4. Le code pour écrire une note spéciale pour le professeur	29
9.5. Et si je préfère utiliser mes propres templates de code ?	30
9.6. C'est lourd d'adopter les templates de code conseillés !	30
10. Injecter automatiquement un template de fichier d'entête pour un "livre"	30
11. Injecter automatiquement un template de fichier d'entête pour un chapitre	32
12. Injecter automatiquement un template de fichier d'entête pour un exercice	33
13. Partager des composants entre des chapitres de thèmes différents	35
14. Partager des composants entre des chapitres d'un même thème	38
15. Partager des composants entre des exercices d'un même chapitre	40
16. Partager des composants entre des livres	41
17. Utilisation avec asciidoctor-diagram	44
Index	46

1. Lien entre AsciiDoc et Asciidocpro

1.1. Qu'est-ce que AsciiDoc ?

AsciiDoc est un langage de balisage léger conçu pour la rédaction de documents, particulièrement adapté pour la documentation technique. Il permet de créer des documents bien structurés à l'aide d'une syntaxe simple et lisible, tout en offrant des fonctionnalités avancées comme la génération automatique de tables des matières, d'index, de bibliographies et bien plus encore.

Si vous recherchez un langage qui vous permet d'écrire rapidement des documentations, des supports de cours, des comptes rendus ou tout autre support écrit, je ne peux que vous conseiller le langage **AsciiDoc**.

La syntaxe AsciiDoc permet aux rédacteurs de se concentrer sur le contenu de leur document plutôt que sur des détails de mise en forme complexes.

Les auteurs peuvent écrire leur contenu en utilisant une syntaxe intuitive et proche du langage naturel, sans avoir à se soucier immédiatement de la manière dont le document sera formaté. AsciiDoc permet ensuite de convertir ce contenu en différents formats de sortie (comme HTML, PDF, etc.) en appliquant automatiquement les règles de mise en forme et de présentation spécifiées.

Cela signifie que les rédacteurs peuvent se concentrer sur l'expression de leurs idées et la structure de leur document, sans être distraits par les détails de mise en forme. Cela peut rendre le processus d'écriture plus fluide et permettre de produire du contenu de manière plus efficace.

1.2. Qu'est-ce qu'Asciidocpro ?

Asciidocpro est un framework qui facilite l'écriture de supports au sein d'IDE tels que ceux proposés par JetBrains ou encore Visual Studio Code. Pour faire simple, c'est une arborescence de dossiers et de fichiers utilisant le langage AsciiDoc. Ces fichiers contiennent une logique qui permet de faciliter et d'automatiser la création de supports écrits avec le langage AsciiDoc.

Vous pouvez écrire autant de supports que vous voulez au sein d'un seul projet Asciidocpro.

1.3. Pourquoi adopter Asciidocpro ?

Lorsqu'un document écrit avec AsciiDoc devient plus ou moins complexe, il est préférable de le découper en parties plus petites. Cela peut être réalisé avec peu d'efforts et facilite grandement ses mises à jour.

Les choses commencent à devenir plus compliquées lorsque vous souhaitez produire des supports avec certains chapitres et pas d'autres, en affichant ou masquant les réponses, etc. Cela devient encore plus problématique lorsqu'il faut intégrer le travail d'un collègue, d'un élève, etc.

Asciidocpro vous donne un cadre de travail qui vous "force" à organiser vos fichiers de façon à faciliter leur exploitation par la suite.

Si vous travaillez à plusieurs sur un support, avec Asciidocpro, il est très facile de regrouper les

parties de chaque collaborateur et de générer un support unique.

Si vous connaissez et utilisez Asciidoc dans un IDE, vous utilisez la prévisualisation. Dans un document basé sur un unique fichier, cela ne pose aucun problème. Par contre, dès que le support est découpé en plusieurs fichiers, il devient compliqué de partager les attributs, d'avoir une table des matières seulement pour la prévisualisation, etc.

Asciidocpro vous permet de lier très facilement les différentes parties d'un support tout en partageant automatiquement le même contexte. Effectivement, Asciidocpro vous permet de déclarer des attributs qui vont être automatiquement injectés dans tous vos fichiers AsciiDoc. Vous les définissez une seule fois et vous n'avez plus qu'à les utiliser.

Asciidocpro vous permet lors de la rédaction :

- de partager des attributs entre les différentes parties de votre support
- de disposer de la prévisualisation d'une table des matières au niveau d'un chapitre, d'un exercice ou d'un livre
- d'ajouter à un document des chapitres et ou des exercices, de les déplacer sans avoir à modifier quoi que ce soit dans ces chapitres ou exercices.
- de définir une durée pour chaque livre, chapitre ou exercice
- de définir un auteur pour chaque livre, chapitre ou exercice
- etc

Asciidocpro vous permet lors de la génération du support final :

- de gérer facilement différents rendus à partir du même support
 - génération d'un support sans les réponses
 - génération du même support avec les réponses
 - génération du même support avec des notes destinées au professeur, formateur
- de configurer un rendu spécifique pour un livre en particulier
- etc

Pour faire très simple, Asciidocpro c'est :

- un cadre qui vous permet d'organiser vos chapitres avec leurs ressources
- un outil capable de grouper des chapitres et / ou des exercices pour en faire un seul support
- un outil qui travaille pour vous de façon transparente en automatisant certaines opérations de rendu
- un outil configurable qui vous permet de personnaliser certains comportements et rendus
- un outil qui améliore la prévisualisation du contenu AsciiDoc dans les IDE.

1.4. La philosophie d'Asciidocpro

Asciidocpro repose sur un principe fondamental : un document est un ensemble composé de

parties élémentaires (des chapitres, des exercices). Chaque partie élémentaire est écrite dans un fichier AsciiDoc.

En réalité, vous n'écrivez que des chapitres et ou des exercices, jamais un support complet.

Une fois ces parties élémentaires écrites, vous les regroupez depuis un fichier "principal" dans l'ordre de votre choix.

Il y a bien des avantages à opter pour une approche par "partie élémentaire" :

- une partie élémentaire (un chapitre, un exercice) peut être intégrée ou non dans un support
- il est facile de travailler sur une partie élémentaire sans l'intégrer immédiatement dans le support final
- une partie élémentaire peut être utilisées dans plusieurs supports. La mise à jour d'une partie est donc répercutée dans tous les supports qui l'intègrent.

Vous pouvez vous concentrer sur la rédaction de vos supports, Asciidocpro gère le reste.

1.5. Comment mettre en place Asciidocpro ?

Il vous suffit de récupérer la structure du projet Asciidocpro et de la placer dans le répertoire de votre choix... et c'est tout.

1.6. Comment utiliser Asciidocpro ?

Pour débiter avec Asciidocpro, il faut commencer par apprendre à [créer un chapitre](#). Si un chapitre prévoit des exercices, vous devez lire comment [créer un exercice](#). Une fois vos chapitres et / ou exercices écrits, vous pouvez lire comment [créer un "livre"](#). Vous pouvez ensuite apprendre à [personnaliser le rendu d'un document](#).

Une fois que vous avez appréhendé ces 4 fondamentaux, vous pouvez passer aux autres fonctionnalités proposées par Asciidocpro. Elles sont toutes présentées dans cette documentation.



Je pars tout de même du principe que vous connaissez AsciiDoc et que vous savez générer un fichier pdf à partir d'un document écrit avec ce langage.

Asciidocpro peut être managé avec AsciidocproM.

AsciidocproM est un logiciel qui exploite un projet Asciidocpro et qui propose des fonctionnalités accessibles via une interface graphique. Voici quelques fonctionnalités couvertes à plus ou moins long terme par ce logiciel :

- automatisation de la mise à jour d'Asciidocpro
- automatisation de la création d'un nouveau chapitre
- automatisation de la création d'un nouvel exercice
- automatisation de la création d'un nouveau support
- création automatisée d'un support à partir d'un dossier de thème

- création automatisée d'un support constitué d'un chapitre et de ses exercices
- création automatisée d'un support à partir de chapitres divers et / ou d'exercices divers
- agrégation des compétences définies au niveau de chaque chapitre et génération d'un récapitulatif
- intégration des différents outils permettant de générer un fichier pdf sans installation supplémentaire
- calcul de la durée totale de réalisation d'un support
- refactorisation automatique suite à modification d'un nom de thème
- refactorisation automatique suite à modification du nom d'un dossier de chapitre ou en cas de déplacement
- refactorisation automatique suite à modification du nom d'un dossier d'exercice ou en cas de déplacement
- synchronisation d'un projet Asciidocpro via gitlab
- versionnement des chapitres et intégration dans les supports publiés
- et encore de nombreuses autres fonctionnalités

2. Asciidoc, AsciiDoctor, Asciidocpro, AsciidocproM, c'est quoi ?

Il ne faut pas confondre les termes suivants :

- asciidoc
- asciidocpro
- asciidocproM
- asciidoctor

Explications :

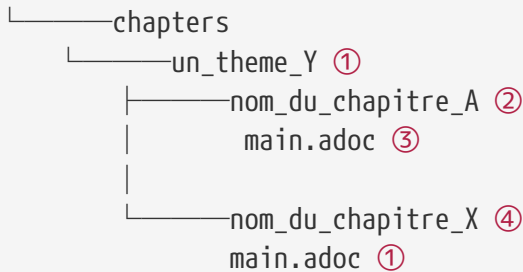
- [AsciiDoc](#) est le langage de balisage qui permet d'écrire des documents
- [AsciiDoctor](#) est un outil qui permet de convertir un document écrit en asciidoc vers des formats tels que pdf, html5, DocBook, epub, etc
- [Asciidocpro](#) est le nom du projet qui fournit un cadre de travail pour écrire des supports. C'est l'objet de la présente documentation.
- [AsciidocproM](#) est le logiciel qui permet de manager un projet Asciidocpro.

3. Ecrire un chapitre

3.1. Où écrire un chapitre ?

Un chapitre doit être écrit dans un fichier nommé `main.adoc`. Ce fichier doit être placé **dans un sous-dossier** du dossier `chapters`. Vous pouvez voir ce sous-dossier comme un dossier de thème. Il vous permet de regrouper des chapitres au sein d'un même dossier.

Voici un exemple de deux chapitres regroupés dans le dossier `un_theme_Y` :



- ① dossier qui regroupe les chapitres appartenant au même sujet / thème
- ② dossier qui contient tous les éléments du chapitre
- ③ fichier `main.adoc` dans lequel écrire le contenu du chapitre
- ④ un autre chapitre qui appartient au même sujet que le chapitre précédent



Si vous utilisez asciidocproM, chaque dossier doit avoir un nom unique, qu'il s'agit d'un dossier de thème ou un dossier de chapitre.

De plus, même si vous n'utilisez pas asciidocproM, utiliser des noms uniques vous permet de rapidement retrouver un chapitre ou un exercice par son nom.

Depuis un IDE JetBrains, appuyez deux fois de suite sur la touche `SHIFT` et vous pourrez faire une recherche dans tout le projet !

3.2. Règles à respecter pour débiter un fichier de chapitre

Le nom du fichier d'un chapitre doit être `main.adoc`. L'identification d'un chapitre est possible grâce au dossier qui le contient. C'est pour cela qu'il est très fortement conseillé d'avoir des noms de dossier unique dans tout le projet asciidoc.

Pour que votre chapitre puisse être intégré dans le fonctionnement d'asciidocpro, vous devez obligatoirement écrire au début de votre fichier `main.adoc` le contenu suivant :

```

:_duration: ①
:_author: ②
[[chapitre_Ici_le_titre_de_mom_chapitre]] ③
= Ici le titre de mom chapitre ④
\include:../../config/application/includes/start_chapter.adoc[] ⑤

```


- ① durée de réalisation (ex : 1h et 20 min; 1h 20m; 1h 20minutes, ...). L'important est de pouvoir distinguer les heures des minutes grâce à la présence de la lettre **h** et de la lettre **m**. Si aucune unité n'est exprimée, alors la durée est considérée comme exprimée en heures. Cet attribut est facultatif et peut être laissé vide ou supprimé.
- ② Auteur du support. Si une valeur est définie, l'auteur configuré par défaut pour tout le projet sera écrasé par la valeur spécifiée. S'il est laissé vide ou s'il est supprimé, l'auteur configuré par défaut sera affiché.
- ③ Identifiant qui peut servir d'ancre. Cet élément est également facultatif mais est très utile pour faire des liens depuis un chapitre vers un autre.
- ④ Le titre du chapitre qui doit être un titre de niveau 0 (soit avec un signe **=**).
- ⑤ inclusion du fichier qui permet d'automatiser tout à un tas de choses. Il faut veiller à bien spécifier le chemin `../../../../config/application/includes/` et à indiquer le bon fichier `start_chapter.adoc`.



Il ne faut surtout pas de ligne vide avant d'avoir inclus le fichier `start_chapter.adoc`.

Si vous ne souhaitez jamais spécifier d'auteur et de durée et que vous ne faites pas de lien entre les chapitres, vous pouvez réduire le contenu au titre et à l'inclusion du fichier `start_chapter.adoc` :

```
= Ici le titre du chapitre
include:../../../../config/application/includes/start_chapter.adoc[]
```

3.3. Bien délimiter le contenu d'un chapitre

3.3.1. Rendre les chapitres atomiques

Un **chapitre** doit être vu comme une unité atomique, c'est-à-dire comme un contenu dont les éléments sont très fortement liés entre eux. Si le fait de séparer ces éléments de contenu ne gêne en rien la compréhension du chapitre, c'est que votre contenu n'est pas atomique.

Il faut voir un chapitre comme une unité qui peut vivre en autonomie. Un chapitre doit pouvoir être communiqué seul sans que cela nuise à sa compréhension.



L'erreur courante est de mettre plusieurs notions dans un même chapitre alors qu'elles pourraient être séparées dans des chapitres distincts.

Mais pourquoi adopter cette démarche d'atomicité des chapitres ?

Le fait d'écrire chaque chapitre comme une unité qui peut vivre en autonomie vous permet de :

- déplacer le dossier du chapitre dans un autre dossier de thème. C'est très pratique lorsque vous vous rendez compte après coup que tel chapitre serait mieux placé dans un autre dossier.
- générer un "livre" avec un contrôle fin sur les chapitres qui le composent.
- faciliter la modification des chemins utilisés depuis un "livre" en cas de déplacement d'un

chapitre ou de modification du nom du dossier de thème ou de chapitre.

3.3.2. Organiser les répertoires d'un dossier de chapitre

Un chapitre est un dossier qui contient au minimum le fichier `main.adoc`.

Mais vous pouvez avoir besoin d'utiliser des images, des fichiers de code, etc, depuis votre fichier `main.adoc`.

Asciidocpro encadre la structure d'un dossier de chapitre. Vous devez adopter les sous-dossiers suivants :

- `images` : dossier qui va stocker les images utilisées dans le chapitre
- `code` : dossier qui va stocker le code utilisé dans le chapitre. Il est très fortement conseillé de créer des sous-dossiers pour regrouper les fichiers de code en fonction des notions abordées dans le chapitre !
- `assets` : dossier qui contient les ressources qui doivent être distribuées aux étudiants. Il est important de regrouper les ressources dans des sous-dossiers afin de mieux contrôler leur diffusion.
- `extras` : dossier contenant des éléments utiles à l'auteur et en rapport de prêt ou de loin avec le chapitre. Le contenu de ce dossier est une ressource pour l'auteur et n'a pas vocation à être diffusé.
- `exercices` : dossier qui va stocker les exercices relatifs au chapitre. Il est impératif de créer un sous-dossier par exercice. Les dossiers à créer dans le dossier d'exercice sont les mêmes que pour le dossier d'un chapitre.

L'avantage de cette organisation est qu'il est facile d'inclure des fichiers car les chemins sont relatifs au fichier `main.adoc` dans lequel vous écrivez votre contenu.

En plus de ces dossiers, vous êtes libre de créer d'autres sous-dossiers en fonction de vos besoins.



Adopter cette structure vous permet de partager facilement des chapitres et de travailler à plusieurs.

Cette structure vous permet de voir un chapitre comme une seule unité déplaçable dans un autre dossier de thème. Cela supprime le couplage qu'il peut y avoir entre les répertoires.

De plus, AsciidocproM exploite la structure d'un projet Asciidocpro. Bien respecter cette structure vous assure une pleine compatibilité.

3.3.3. Utiliser des images dans un chapitre

Les images d'un chapitre sont à placer dans le dossier `images` du chapitre concerné. Si cette contrainte n'est pas respectée, Asciidocpro ne sera pas en mesure de résoudre les chemins des images lorsque vous créerez un "livre".

Dans votre fichier `main.adoc`, pour insérer une image, vous n'avez qu'à écrire :

```
image::images/nom_du_fichier_image.png[]
```

4. Ecrire un exercice

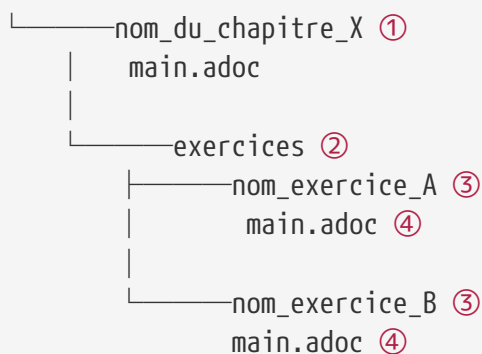
4.1. Où écrire un exercice ?

Les exercices d'un chapitre doivent être créés dans un sous-dossier du chapitre nommé `exercices`.

Chaque `exercice` doit avoir son propre dossier dans le dossier `exercices`.

Dans le dossier de l'exercice, créez un fichier nommé `main.adoc`.

Voici un exemple de chapitre qui contient deux exercices :



- ① dossier de chapitre qui contient des exercices
- ② dossier dans lequel placer tous les exercices du chapitre
- ③ dossier contenant un et un seul exercice
- ④ fichier dans lequel écrire le contenu de l'exercice

4.2. Règles à respecter pour débiter un fichier d'exercice

Les règles à respecter pour débiter un fichier d'exercice sont les mêmes que celle d'un fichier de chapitre à une nuance prêt.

Le nom du fichier d'un exercice doit être `main.adoc`. L'identification d'un exercice est possible grâce au dossier qui le contient.

Pour que votre exercice puisse être intégré dans le fonctionnement d'Asciidocpro, vous devez obligatoirement écrire au début de votre fichier `main.adoc` le contenu suivant :

```
:_duration: ①
:_author: ②
[[exercice_ici_le_titre_de_lexercice]] ③
```

```
= Ici le titre de l'exercice ④
\include:../../../../config/application/includes/start_exercise.adoc[] ⑤
```

- ① durée de réalisation (ex : 1h et 20 min; 1h 20m; 1h 20minutes, ...). L'important est de pouvoir distinguer les heures des minutes grâce à la présence de la lettre **h** et de la lettre **m**. Si aucune unité n'est exprimée, la durée est considérée comme exprimée en heures. Cet attribut est facultatif et peut être laissé vide ou supprimé.
- ② Auteur du support. Si une valeur est définie, l'auteur configuré par défaut pour tout le projet sera écrasé par la valeur spécifiée. S'il est laissé vide ou s'il est supprimé, l'auteur configuré par défaut sera affiché.
- ③ Identifiant qui peut servir d'ancre.
- ④ Le titre de l'exercice qui doit être un titre de niveau 0 (soit avec un signe **=**).
- ⑤ inclusion du fichier qui permet d'automatiser tout à un tas de choses. Il faut veiller à bien spécifier le chemin `../../../../config/application/includes/` et à indiquer le bon fichier `start_exercise.adoc`. **Attention, le chemin n'est pas identique à celui utilisé depuis un chapitre**



Il ne faut surtout pas de ligne vide avant d'avoir inclus le fichier `start_chapter.adoc`.

Si vous ne souhaitez jamais spécifier d'auteur et de durée et que vous ne faites pas de lien vers des exercices, vous pouvez réduire le contenu au titre et à l'inclusion du fichier `start_exercise.adoc` :

```
= Ici le titre de l'exercice
include:../../../../config/application/includes/start_exercise.adoc[]
```

4.3. Organiser les répertoires d'un dossier d'exercice

L'organisation est pratiquement identique à celle d'un dossier de chapitre.

Un exercice est un dossier qui contient au minimum le fichier `main.adoc`.

Mais vous pouvez avoir besoin d'utiliser des images, des fichiers de code, etc, depuis votre fichier `main.adoc`.

Asciidocpro encadre la structure d'un dossier de chapitre. Vous devez adopter les sous-dossiers suivants :

- `images` : dossier qui va stocker les images utilisées dans le chapitre
- `code` : dossier qui va stocker le code utilisé dans le chapitre. Il est très fortement conseillé de créer des sous-dossiers pour regrouper les fichiers de code en fonction des notions abordées dans le chapitre !
- `assets` : dossier qui contient les ressources qui doivent être distribuées aux étudiants. Il est important de regrouper les ressources dans des sous-dossiers afin de mieux contrôler leur diffusion.

- **extras** : dossier contenant des éléments utiles à l'auteur et en rapport de prêt ou de loin avec le chapitre. Le contenu de ce dossier est une ressource pour l'auteur et n'a pas vocation à être diffusé.

4.4. Utiliser des images dans un exercice

Les images d'un exercice sont à placer dans le dossier **images** de l'exercice concerné. Si cette contrainte n'est pas respectée, Asciidocpro ne sera pas en mesure de résoudre les chemins des images lorsque vous créerez un "livre".

Dans votre fichier **main.adoc**, pour insérer une image, vous n'avez qu'à écrire :

```
image::images/nom_du_fichier_image.png[]
```

5. Créer un "livre"

5.1. C'est quoi un livre dans Asciidocpro ?

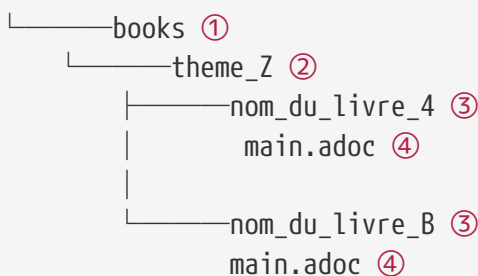
Un **livre** doit être vu comme le fait de construire un support à partir d'un seul ou de plusieurs chapitres, sans ou avec des exercices.

C'est là que le fait d'avoir écrit des **chapitres atomiques** va vous être très utile. Effectivement, il est facile de déterminer les différents chapitres et / ou exercices qui doivent composer votre support.

5.2. Où écrire un livre ?

Un livre doit être écrit dans un fichier nommé **main.adoc**. Ce fichier doit être placé dans un sous-dossier du dossier **books**. Vous pouvez voir ce sous-dossier comme un dossier de thème. Il vous permet de regrouper des livres au sein d'un même dossier.

Voici un exemple de deux livres regroupés dans le même dossier "theme_Z" :



- ① dossier qui contient tous les "livres"
- ② dossier qui permet de regrouper tous les livres relatifs au thème **theme_Z**
- ③ dossier qui contient le livre
- ④ fichier qui contient référence les parties du "livre"

5.3. Règles à respecter pour débiter un fichier de livre



Ces règles ne sont pas applicables lorsque vous souhaitez créer un livre à partir d'un seul chapitre (avec ou sans exercices).

Le nom du fichier d'un "livre" doit être `main.adoc`. L'identification d'un livre est possible grâce au dossier qui le contient. C'est pour cela qu'il est très fortement conseillé d'avoir des noms de dossier unique dans tout le projet asciidoc.

Pour que votre "livre" puisse être intégré dans le fonctionnement d'Asciidocpro, vous devez obligatoirement écrire au début de votre fichier `main.adoc` le contenu suivant :

```
:_duration: ①
:_author: ②
[[chapitre_chapitre_Ici_le_titre_de_mom_chapitre]] ③
= Ici le titre du livre ④
\include:../../../../../config/application/includes/start_book.adoc[] ⑤
```

- ① durée totale de réalisation du livre (ex : 1h et 20 min; 1h 20m; 1h 20minutes, ...). L'important est de pouvoir distinguer les heures des minutes grâce à la présence de la lettre `h` et de la lettre `m`. Si aucune unité n'est exprimée, alors la durée est considérée comme exprimée en heures. Cet attribut est facultatif et peut être laissé vide ou supprimé.
- ② Auteur du support. Si une valeur est définie, l'auteur configuré par défaut pour tout le projet sera écrasé par la valeur spécifiée. S'il est laissé vide ou s'il est supprimé, l'auteur configuré par défaut sera affiché.
- ③ Identifiant qui peut servir d'ancre. Cet élément est également facultatif.
- ④ Le titre du "livre" qui doit être un titre de niveau 0 (soit avec un signe `=`).
- ⑤ inclusion du fichier qui permet d'automatiser tout à un tas de choses. Il faut veiller à bien spécifier le chemin `../../../../../config/application/includes/` et à indiquer le bon fichier `start_book.adoc`.



Il ne faut surtout pas de ligne vide avant d'avoir inclus le fichier `start_book.adoc`.

Si vous ne souhaitez jamais spécifier d'auteur et de durée et que vous ne faites pas de lien vers des exercices, vous pouvez réduire le contenu au titre et à l'inclusion du fichier `start_exercise.adoc` :

```
= Ici le titre du livre
include:../../../../../config/application/includes/start_book.adoc[]
```

5.4. Les différents "livres" qu'il est possible de créer

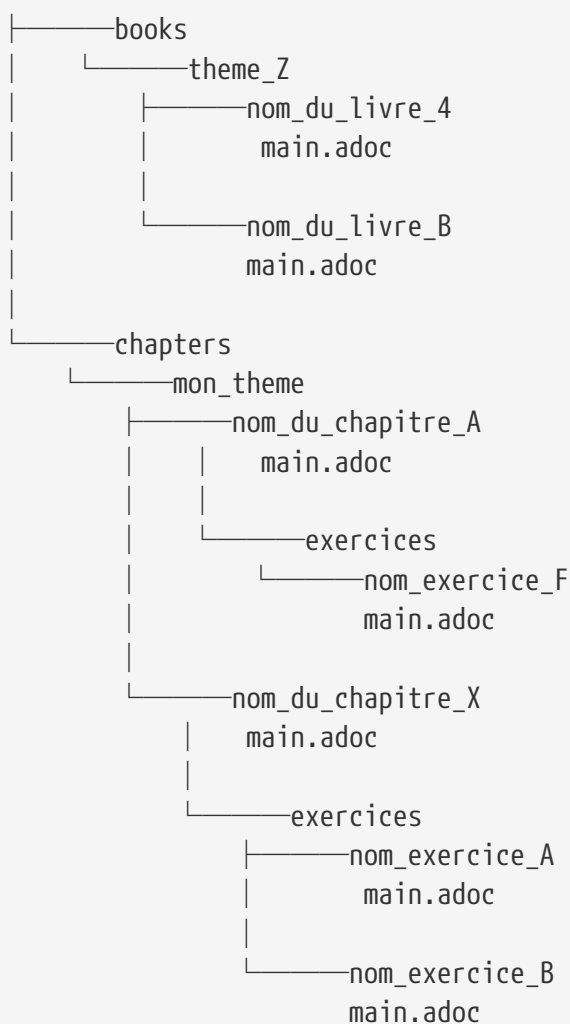
Il est possible de créer :

- un support contenant plusieurs chapitres sans ou avec exercices

- un support ne contenant que des exercices
- un support contenant un seul chapitre
- un support contenant un seul chapitre avec ses exercices

5.4.1. La structure prise en exemple

Voici un exemple de structure d'un projet Asciidocpro qui va nous servir de support pour créer nos différents livres :



5.4.2. Créer un livre avec plusieurs chapitres

Créer un "livre" composé de plusieurs chapitres est le cas le plus courant.

Tout d'abord, il faut préparer le début du fichier `main.adoc` tel que défini dans les [règles précédentes](#).

Fichier `books/theme_Z/nom_du_livre_4/main.adoc` :

```

:_duration:
:_author:
[[book_Asciidocpro]]
  
```

```
= AsciiDocpro
include:../../config/application/includes/start_book.adoc[]
```

Ensuite, il faut indiquer le répertoire du premier chapitre que l'on souhaite ajouter.

Dans le cas présent, nous voulons ajouter le chapitre correspondant au dossier `nom_du_chapitre_X`. Le chemin à utiliser est celui qui part de la racine du projet AsciiDocpro jusqu'au dossier du chapitre à inclure, c'est-à-dire `chapters/mon_theme/nom_du_chapitre_X`.

Fichier `books/theme_Z/nom_du_livre_4/main.adoc` :

```
:_duration:
:_author:
[[book_AsciiDocpro]]
= AsciiDocpro
include:../../config/application/includes/start_book.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ①
\include:../../config/application/includes/add_chapter.adoc[] ②
```

① chemin vers le chapitre à ajouter au "livre" précisé dans l'attribut `_chapter_folder_path`

② inclusion du fichier qui permet d'automatiser tout à un tas de choses. Il faut veiller à bien spécifier le chemin `../../config/application/includes/` et à indiquer le bon fichier `add_chapter.adoc`.

Pour ajouter un second chapitre, il faut procéder de la même façon en adaptant la valeur de l'attribut `_chapter_folder_path`.

Ajoutons le chapitre `nom_exercice_B` après le chapitre précédemment inséré.

Fichier `books/theme_Z/nom_du_livre_4/main.adoc` :

```
:_duration:
:_author:
[[book_AsciiDocpro]]
= AsciiDocpro
include:../../config/application/includes/start_book.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X //
include:../../config/application/includes/add_chapter.adoc[]

//ajout d'un second chapitre
:_chapter_folder_path: chapters/mon_theme/nom_exercice_B ① ②
\include:../../config/application/includes/add_chapter.adoc[] ③
```

① le chemin du chapitre à ajouter est spécifié dans l'attribut `_chapter_folder_path`.

- ② n'importe quel chapitre peut être inséré, même s'il ne fait pas parti du même thème.
- ③ l'inclusion du fichier `add_chapter.adoc` permet d'insérer le chapitre dans le "livre".

Vous pouvez ajouter autant de chapitres que vous le souhaitez.

Une fois satisfait, vous pouvez générer le fichier soit avec la commande d'[asciidocctor-pdf](#), soit via le plugin Asciidoc préalablement installé dans votre IDE.

5.4.3. Un livre avec des chapitres et des exercices

Si vous souhaitez ajouter les exercices aux chapitres déjà inclus, c'est très simple. Il suffit de spécifier le nom du dossier de l'exercice à inclure dans l'attribut `_exercise_folder_name`. **Attention, je n'ai pas parlé de chemin mais seulement du nom du dossier de l'exercice**



Pour inclure un exercice, il faut avoir préalablement inclus le chapitre qui contient cet exercice.

Asciidocpro va automatiquement "lier" l'exercice au chapitre précédemment ajouté.

Voici un exemple où les deux exercices du chapitre `nom_du_chapitre_X` sont ajoutés ainsi que l'exercice du chapitre `nom_du_chapitre_A`:

```
:_duration:
:_author:
[[book_Asciidocpro]]
= Asciidocpro
include:../../../../config/application/includes/start_book.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X //
include:../../../../config/application/includes/add_chapter.adoc[]

//ajout du premier exercice du chapitre ①
:_exercise_folder_name: nom_exercice_B ②
\include:../../../../config/application/includes/add_exercise.adoc[] ③

//ajout du second exercice du chapitre ④
:_exercise_folder_name: nom_exercice_A
include:../../../../config/application/includes/add_exercise.adoc[]

//ajout d'un second chapitre
:_chapter_folder_path: chapters/mon_theme/nom_exercice_B
include:../../../../config/application/includes/add_chapter.adoc[]

//ajout de l'exercice dans le chapitre précédemment inséré ⑤
:_exercise_folder_name: nom_exercice_F
include:../../../../config/application/includes/add_exercise.adoc[]
```

- ① L'ajout d'un exercice se fait après l'ajout d'un chapitre
- ② L'attribut `_exercice_folder_name` permet d'indiquer le nom du dossier d'exercice à ajouter. Ce dossier sera automatiquement recherché dans le dossier du dernier chapitre inséré.
- ③ inclusion du fichier qui permet d'automatiser la liaison de l'exercice au chapitre précédemment ajouté
- ④ Ajout d'un second exercice avec la même logique : le nom du dossier d'exercice est précisé dans l'attribut `_exercice_folder_name` puis le fichier `add_exercice.adoc` est inclus pour faire la liaison.
- ⑤ Ajout d'un exercice qui sera lié au dernier chapitre inséré.



Il faut veiller à utiliser une valeur pour `_exercice_folder_name` qui correspond à un dossier d'exercice du dernier chapitre ajouté.

5.4.4. Un livre qui ne contient que des exercices

Si vous souhaitez créer un support qui ne contient que des exercices sans leur chapitre respectif, il vous commence par créer le début du fichier "livre" `main.adoc` conformément aux [règles applicables pour débiter un livre](#).

Fichier `books/theme_Z/livre_d_exercices/main.adoc` :

```
:_duration:
:_author:
[[book_mon_livre_dexercices]]
= Mon livre d'exercices
include:../../../config/application/includes/start_book.adoc[]
```

Une fois la base en place, j'indique le chemin du chapitre depuis lequel proviennent les exercices que je souhaite ajouter avec l'attribut `_chapter_folder_path`. Enfin, je précise le nom de chaque dossier d'exercice de ce chapitre avec l'attribut `_exercice_folder_name` suivi de la ligne d'ajout de l'exercice :

```
:_duration:
:_author:
[[book_mon_livre_dexercices]]
= Mon livre d'exercices
include:../../../config/application/includes/start_book.adoc[]

//indication du chapitre qui contient les exercices à ajouter
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ①
//ajout du premier exercice du chapitre
:_exercice_folder_name: nom_exercice_B ②
\include:../../../config/application/includes/add_exercice.adoc[] ③
//ajout du second exercice du chapitre
:_exercice_folder_name: nom_exercice_A ④
\include:../../../config/application/includes/add_exercice.adoc[] ⑤
```

```
//indication d'un autre chapitre qui contient les exercices à ajouter
:_chapter_folder_path: chapters/mon_theme/nom_exercice_B ⑥
//ajout de l'exercice dans le chapitre
:_exercise_folder_name: nom_exercice_F ⑦
\include:../../../../../config/application/includes/add_exercice.adoc[] ⑧
```

- ① précision du chemin du dossier de chapitre depuis lequel proviennent les exercices à inclure
- ② précision du nom dossier d'exercice à utiliser. Ce dossier sera recherché dans le dossier spécifié dans le dernier `_chapter_folder_path`.
- ③ inclusion du fichier qui se charge d'injecter le chapitre dans le "livre"
- ④ nom du dossier du second exercice à injecter. Ce dossier sera recherché dans le dossier spécifié dans le dernier `_chapter_folder_path`.
- ⑤ inclusion magique de l'exercice
- ⑥ On change de chapitre pour y puiser les exercices à insérer
- ⑦ nom du dossier d'exercice du dernier chapitre spécifié dans `_chapter_folder_path`
- ⑧ inclusion magique de l'exercice

5.4.5. Un livre avec un seul chapitre



Ce type de livre est particulier car il n'est pas concerné par les [règles à respecter pour débiter un livre](#).

Si vous devez créer un "livre" qui ne contient qu'un seul chapitre, vous devez partir d'un fichier `main.adoc` totalement vide.

Ensuite, vous devez préciser le chemin du dossier du chapitre à ajouter dans l'attribut `_chapter_folder_path` puis inclure le fichier `add_chapter.adoc`.

Voici un exemple d'un "livre" qui ne contient qu'un seul chapitre :

Fichier `books/theme_Z/livre_avec_un_seul_chapitre/main.adoc` :

```
//il n'y a aucun contenu avant cette ligne !
:_chapter_folder_path: chapters/mon_sujet/nom_du_chapitre_A
include:../../../../../config/application/includes/add_chapter.adoc[]
```



Dans ce mode, vous ne pouvez pas ajouter d'autres chapitres.

Si vous souhaitez faire évoluer votre support en ajoutant des chapitres, il vous faudra rajouter les [règles à respecter pour débiter un livre](#).

5.4.6. Un livre avec un seul chapitre et ses exercices

Si vous souhaitez créer un support contenant un chapitre et tout ou partie de ses exercices, le processus est le même que pour un [livre avec un seul chapitre](#). C'est-à-dire qu'il ne faut pas

appliquer les [règles à respecter pour débiter un livre](#) et partir d'un fichier `main.adoc` totalement vide.

Ensuite, vous devez préciser le chemin du dossier du chapitre à ajouter dans l'attribut `_chapter_folder_path` puis inclure le fichier `add_chapter.adoc`. Enfin, il faut préciser le nom du dossier de chaque exercice de ce chapitre puis l'inclure avec le fichier `add_exercise.adoc`.

Voici un exemple d'un "livre" qui ne contient qu'un seul chapitre et ses exercices :

Fichier `books/theme_Z/livre_avec_un_seul_chapitre_et_des_exercices/main.adoc` :

```
//ajout du seul chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ①
\include:../../../../config/application/includes/add_chapter.adoc[] ②

//ajout du premier exercice du chapitre
:_exercise_folder_name: nom_exercice_B ③
\include:../../../../config/application/includes/add_exercise.adoc[] ④

//ajout du second exercice du chapitre ⑤
:_exercise_folder_name: nom_exercice_A
include:../../../../config/application/includes/add_exercise.adoc[]
```

- ① Le chemin du chapitre est spécifié via l'attribut `_chapter_folder_path`
- ② le chapitre ciblé est injecté dans le "livre" L'ajout d'un exercice se fait après l'ajout d'un chapitre
- ③ L'attribut `_exercise_folder_name` permet d'indiquer le nom du dossier d'exercice à ajouter. Ce dossier sera automatiquement recherché dans le dossier du chapitre inséré.
- ④ inclusion du fichier qui permet d'automatiser la liaison de l'exercice au chapitre précédemment ajouté
- ⑤ Ajout d'un second exercice avec la même logique : le nom du dossier d'exercice est précisé dans l'attribut `_exercise_folder_name` puis le fichier `add_exercise.adoc` est inclus pour faire la liaison.

5.5. Marquer un livre comme terminé

Lorsque vous avez terminé d'ajouter vos chapitres et ou vos exercices, vous devez indiquer que le livre est terminé avec l'instruction suivante :

```
//on force le saut de page
<<<
:leveloffset: 0
[index]
== Index
//end eval _show_index
```

Si vous ommettez cette ligne, cela ne gêne en rien la génération du document final. Cependant, vous ne pourrez rendre l'[index](#).

6. Personnaliser le rendu d'un document

6.1. Qu'est-il possible de personnaliser ?

En utilisant les attributs à votre disposition, vous pouvez modifier le rendu du document dans votre IDE et dans le fichier pdf généré.

Vous pouvez personnaliser les éléments suivants :

- l'ajout d'un préfixe aux titres des chapitres
- le nom de l'auteur à utiliser par défaut dans tous les supports
- l'activation de plantuml
- le titre de la partie d'un chapitre qui contient les exercices
- le nom du fichier de template d'entête d'un "livre" à insérer après la table des matières
- le nom du fichier de template d'entête d'un chapitre à insérer après le titre du chapitre
- le nom du fichier de template d'entête d'un exercice à insérer après le titre d'un exercice
- le nom du fichier de thème qui permet de personnaliser le rendu pdf utilisé par défaut
- l'affichage de l'auteur sous chaque chapitre et exercice
- l'affichage des réponses
- l'affichage des templates d'entête
- l'affichage des notes du professeur
- l'affichage des numéros de titre
- l'affichage de la table des matières
- les niveaux de titres à reprendre dans la table des matières
- le nom à utiliser au-dessus de la table des matières

Vous pouvez configurer les attributs :

- **au niveau global** : les valeurs des attributs sont applicables à tous les livres, chapitres et exercices.
- **au niveau d'un "livre"** : les valeurs sont applicables au "livre" concerné. Toute valeur définie au niveau locale écrase celle définie au niveau global.

6.2. Configurer les attributs d'application au niveau global

Les **attributs d'application** sont des attributs qui impactent le fonctionnement d'Asciidocpro, notamment le rendu du fichier pdf.



Pour spécifier une valeur personnalisée pour un attribut donné, vous devez

définir cet attribut dans le fichier `config/custom/attributes/user_application_attributes.adoc`.



Les attributs définis dans `config/custom/attributes/user_application_attributes.adoc` sont appliqués à tous les livres, chapitres et exercices du projet AsciiDocpro.

Voici un exemple qui modifie la valeur par défaut du label de la table des matières :

fichier `config/custom/attributes/user_application_attributes.adoc`

```
:_user_toc_title: Sommaire
```

Maintenant, les tables des matières auront toutes le label "Sommaire" à la place de la valeur par défaut "Table des matières".

Voici un autre exemple qui en plus de modifier le label de la table des matières configure le nom de l'auteur par défaut :

```
:_user_toc_title: Sommaire
:_user_default_author: Chuck Norris
```



Aucune ligne vide ne doit être laissée dans le fichier `config/custom/attributes/user_application_attributes.adoc`

Laisser une ligne vide avant, entre ou après les attributs fera échouer le rendu du document.

Voici la liste de tous les attributs AsciiDocpro que vous pouvez utiliser pour personnaliser le comportement d'AsciiDocpro. Les attributs sont classés par ordre alphabétique, avec l'explication de leur utilité. Pour chacun d'eux, leur valeur par défaut est précisée. Le fichier `config/custom/attributes/user_application_attributes.adoc` ne devrait contenir que les attributs dont les valeurs par défaut ne vous conviennent pas.

Liste des attributs d'application :

```
// -----
// Préfix automatique avant chaque titre de chapitre
//
// A laisser vide pour qu'il n'y ait aucun préfixe.
// -----
:_user_chapter_label:
//
// -----
// Auteur par défaut
//
// Auteur à utiliser lorsque aucun auteur n'est spécifié au niveau d'un livre,
```

```
//d'un chapitre ou d'un exercice.
// -----
:_user_default_author:
//
// -----
// Activation du cache des diagrammes asciidoc-diagram
// https://docs.asciidoctor.org/diagram-extension/latest/generate/#diagram_caching
// 1 pour activer le cache, 0 pour le désactiver
// -----
:_user_enable_cache_diagrams: 1
//
// -----
// Titre de la partie qui regroupe des exercices qui suivent un chapitre
//
// Une valeur doit être spécifiée
// -----
:_user_exercise_title: Exercices
//
// -----
// Nom du fichier asciidoc qui peut être inséré au début de chaque livre, après
// la table des matières.
//
// Aucune valeur par défaut
//
// Si la valeur est définie, le fichier doit être placé dans le répertoire
config/custom/templates/
// -----
:_user_header_book_template_filename:
//
// -----
// Nom du fichier asciidoc de template d'entête pouvant être injecté automatiquement
// sous le titre du chapitre. A laisser vide si vous n'en avez pas besoin.
//
// Aucune valeur par défaut
//
// Si la valeur est définie, le fichier doit être placé dans le répertoire
config/custom/templates/
// -----
:_user_header_chapter_template_filename:
//
// -----
// Nom du fichier asciidoc de template d'entête pouvant être injecté
// automatiquement sous le titre d'un exercice.
//
// Aucune valeur par défaut
//
// Si la valeur est définie, le fichier doit être placé dans le répertoire
config/custom/templates/
// -----
:_user_header_exercise_template_filename:
//
```

```
//-----
// Nom du fichier de thème qui personnalise le rendu du fichier pdf
//
//Le fichier de thème doit être placé dans le dossier config/custom/themes/. Son nom
doit avoir un suffixe *-theme.yml. (ex : mon-joli-style-theme.yml)
//
// Aucune valeur par défaut (le thème par défaut d'Asciidoctor-pdf est utilisé)
//-----
:_user_pdf_theme_filename:
//
// -----
// Affichage de l'auteur sous chaque chapitre et exercice
//
// 1 pour afficher l'auteur, 0 pour le masquer
//
// Valeur par défaut : 0
// -----
:_user_show_author: 0
//
// -----
// Affichage des réponses (à la condition d'utiliser le pattern de la documentation)
//
// 1 pour afficher les réponses, 0 pour les masquer
//
// Valeur par défaut : 1
// -----
:_user_show_correction: 1
//
// -----
// Afficher les templates d'entête
//
// 1 pour afficher, 0 pour masquer
//
// Valeur par défaut : 0
// -----
:_user_show_header_templates: 0
//
// -----
// Afficher l'index
//
// 1 pour afficher, 0 pour masquer
//
// Valeur par défaut : 0
// -----
:_user_show_index: 1
//
// -----
// Affichage des notes du professeur (à la condition d'utiliser le pattern de la
documentation)
//
// 1 pour afficher, 0 pour les masquer
```



```
//
// Valeur par défaut : 1
// -----
:_user_show_note_prof: 1
//
// -----
// Affichage de la numérotation des titres
//
// 1 pour afficher, 0 pour masquer
//
// Valeur par défaut : 1
// -----
:_user_show_title_numbers: 1
//
// -----
// Afficher la table des matières (table of content)
//
// 1 pour afficher, 0 pour masquer
//
// Valeur par défaut : 1
// -----
:_user_show_toc: 1
//
// -----
// Numérotation des titres jusqu'au niveau N
//
// Valeur utilisable : de 1 à 5 (ex avec la valeur 2 : les titres de section des
niveaux
//3 à 5 ne sont pas numérotés)
//
// Valeur par défaut : 5 (tous les niveaux de titre sont numérotés)
// -----
:_user_title_level_number: 5
//
// -----
// Niveaux de titre à afficher dans la table des matières
//
// Valeur : de 1 à 5
//
// Valeur par défaut : 5 (tous les niveaux de titre sont repris dans la table des
matières
// -----
:_user_toc_levels: 5
//
// -----
// Titre de la table des matières
//
// Valeur par défaut : Table des matières
// -----
:_user_toc_title: Table des matières
```

J'insiste fortement sur le fait que laisser une ligne vide dans le fichier `config/custom/attributes/user_application_attributes.adoc` avant, entre ou après les attributs fera échouer le rendu du document.

6.3. Configurer les attributs au niveau d'un "livre"

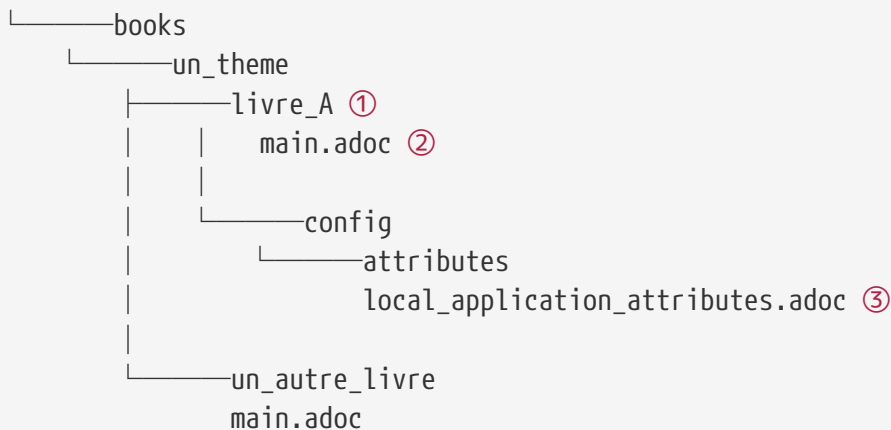
Si vous souhaitez appliquer des valeurs d'attribut spécifiquement à un livre, c'est possible. C'est très pratique pour gérer plus finement le rendu d'un livre.

Imaginez que les réponses soient configurées pour être affichées par défaut (donc au niveau global). Cependant, vous souhaitez qu'elles soient masquées pour un "livre" en particulier.

Cela se fait en 3 étapes :

1. Commencez par créer le répertoire `config/attributes/` dans le dossier qui contient le fichier `main.adoc` de votre "livre". Ensuite créez un fichier nommé `local_application_attributes.adoc`.

Exemple :



- ① dossier du livre pour lequel une personnalisation des attributs est voulue
- ② fichier qui référence les chapitres et ou les exercices qui constituent le "livre"
- ③ fichier `local_application_attributes.adoc` qui contient les valeurs personnalisées des attributs. Ce fichier est placé dans le répertoire `config/attributes/` qui a été créé pour l'occasion dans le répertoire du livre.

2. Votre fichier `local_application_attributes.adoc` est créé et placé au bon endroit, vous pouvez y écrire la valeur de l'attribut `_user_show_correction` à 0 :

```

:_user_show_correction: 0
//et ici tous les attributs de votre choix
  
```



Tous les attributs utilisables au niveau global peuvent être redéfinis au niveau local.



Le fichier `local_application_attributes.adoc` ne doit contenir aucune ligne

vide, y compris à la fin du fichier.

3. A ce niveau, les attributs définis au niveau local ne sont pas encore pris en compte. Pour que cela soit le cas, il faut déclarer l'attribut `_use_local_application_attributes` avant d'inclure le fichier `start_book.adoc` :

```
:_use_local_application_attributes: ①
[[book_titre_de_mon_livre]]
= Titre de mon livre
include:../../../../config/application/includes/start_book.adoc[]
```

- ① la simple déclaration de l'attribut `_use_local_application_attributes` indique à Asciidocpro qu'il doit utiliser le fichier `config/attributes/local_application_attributes.adoc`. Cet attribut doit être déclaré avant d'inclure le fichier `start_book.adoc`.



Si le fichier `local_application_attributes.adoc` n'existe pas alors que l'attribut `_use_local_application_attributes` est déclaré, la génération du pdf échouera en partie.

Si vous ne souhaitez pas appliquer les valeurs personnalisées, il suffit de commenter la ligne :

```
//:_use_local_application_attributes: ①
[[book_titre_de_mon_livre]]
= Titre de mon livre
\include
```

- ① désactivation de la configuration locale (la suppression de la ligne fonctionne également mais laisser la ligne en commentaire indique qu'une configuration locale existe)



Il n'est pas prévu la possibilité de redéfinir les attributs au niveau d'un chapitre ou d'un exercice afin de garder la cohérence entre des livres qui utiliseraient un même chapitre ou exercice.

7. Utiliser ses propres attributs

7.1. Définir des attributs personnalisés au niveau global

Asciidocpro prévoit des **attributs d'application prédéfinis**, c'est-à-dire des attributs dont chaque nom est imposé par Asciidocpro. Ces attributs permettent de configurer le fonctionnement d'Asciidocpro et d'agir sur le rendu des documents générés.

En dehors de ces attributs qui agissent sur l'application, vous pouvez avoir besoin d'utiliser vos propres attributs et que ceux-ci soient utilisables dans tous vos chapitres et exercices.

De tels attributs doivent être déclarés dans le fichier `config/custom/attributes/user_custom_attributes.adoc`.

Imaginons que plusieurs chapitres nécessitent de mentionner l'année scolaire courante. Chaque année, cette valeur doit être mise à jour. Il faut modifier la valeur dans tous les chapitres qui l'utilisent.

Cela peut être évité en utilisant un attribut personnalisé.

Dans le fichier `config/custom/attributes/user_custom_attributes.adoc`, déclarons l'attribut avec le nom de notre choix :

```
:re_annee_scolaire: 2023-2024 ①
```

① le nom de l'attribut est totalement libre. Je l'ai préfixé par mes initiale `re` afin de ne pas rentrer en conflit avec un autre attribut qui aurait le nom `annee_scolaire`.

Il est très important de ne laisser aucune ligne vide dans le fichier `config/custom/attributes/user_custom_attributes.adoc` sans quoi le rendu du document échouera.

Maintenant, depuis n'importe quel fichier de chapitre ou d'exercice, vous pouvez utiliser l'attribut de la façon suivante :

```
Cette année scolaire {re_annee_scolaire} est une année particulièrement chargée...
```

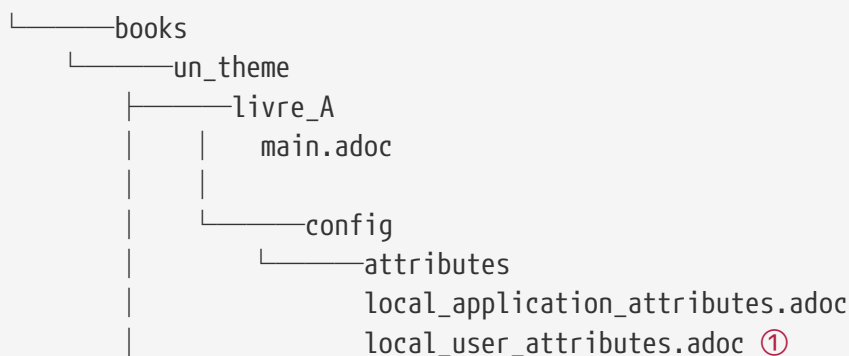
Vous pouvez utiliser autant d'attributs que vous voulez.

7.2. Utiliser des attributs définis au niveau d'un livre

Si vous souhaitez écraser, pour un livre donné, la valeur d'un ou de plusieurs attributs définis globalement dans `config/custom/attributes/user_custom_attributes.adoc`, vous devez le faire dans un fichier respectant le chemin suivant :

`config/attributes/local_custom_attributes.adoc`. Ce chemin doit être relatif au "livre" concerné.

Voici un exemple de "livre" qui prévoit des attributs utilisateurs :



① fichier qui contient les attributs et qui est placé dans `config/attributes/` du livre concerné

livre_A

Le fichier `local_custom_attributes.adoc` ne doit pas contenir de ligne vide, y compris à la fin du fichier. Soyez prudent, car la simple création d'un fichier vide crée une ligne vide dans ce dernier.

Une fois que le fichier est créée (et qu'il ne contient aucune ligne vide), il faut indiquer à AsciiDocpro qu'il doit l'utiliser. Cela se fait en déclarant l'attribut `_use_local_custom_attributes` au niveau d'un "livre" et avant d'inclure le fichier `start_chapter.adoc` :

```
:_use_local_application_attributes:
:_use_local_custom_attributes: ①
[[book_titre_de_mon_super_livre]]
= Titre de mon super livre
include:../../config/application/includes/start_book.adoc[]
```

① AsciiDocpro sait qu'il doit charger le fichier `config/attributes/local_custom_attributes.adoc` du "livre".



Si le fichier `local_custom_attributes.adoc` n'existe pas alors que l'attribut `_use_local_custom_attributes` est déclaré, la génération du pdf échouera en partie.

Si vous ne souhaitez pas appliquer les valeurs personnalisées, il suffit de commenter la ligne :

```
:_use_local_application_attributes:
//:_use_local_custom_attributes: ①
[[book_titre_de_mon_super_livre]]
= Titre de mon super livre
include:../../config/application/includes/start_book.adoc[]
```

① désactivation de la configuration locale (la suppression de la ligne fonctionne également mais laisser la ligne en commentaire indique qu'une configuration locale existe)



Le fichier `local_custom_attributes.adoc` peut tout aussi bien être utilisé pour écraser la valeur d'un attribut personnalisé déclaré au niveau global que pour créer un attribut qui ne sera utilisé que par le "livre" concerné.

8. Utiliser un thème personnalisé

AsciiDocpro permet d'ajouter des thèmes personnalisés si vous ne souhaitez pas utiliser le thème pdf par défaut.

Cela se fait en deux temps :

1. Déposez votre thème personnalisé dans le dossier `config/custom/themes/`. Attention, le nom du fichier de thème doit se terminer par `-theme.yml`
2. Dans le fichier `config/custom/attributes/user_application_attributes.adoc`, configurez l'attribut `_user_pdf_theme_filename` en lui affectant le nom du fichier du thème placé dans le répertoire

précédent

```
// ... d'autres attributs
//...
// fichier de thème à utiliser pour tous les documents
:_user_pdf_theme_filename: mon-propre-theme.yml
```

En utilisant la [personnalisation des attributs au niveau d'un livre](#), vous pouvez définir un fichier de thème pour un livre en particulier.

Si vous souhaitez créer un thème Quelques liens utiles :

- documentation sur les thèmes AsciiDoctor-pdf : <https://docs.asciidoctor.org/pdf-converter/latest/theme/>
- thème de base utilisé par asciidoctor-pdf : <https://github.com/asciidoctor/asciidoctor-pdf/blob/main/data/themes/base-theme.yml>
- thème par défaut utilisé par asciidoctor-pdf : <https://github.com/asciidoctor/asciidoctor-pdf/blob/main/data/themes/default-theme.yml>

9. Les templates de code qu'il faut adopter

9.1. Qu'est-ce qu'un template de code ?

Un **template de code** est un bloc de lignes AsciiDoc dont l'adoption est très fortement recommandée. Adopter les templates de code abordé dans ce chapitre vous permet de profiter de toutes les fonctionnalités de rendu offert par Asciidocpro.

9.2. Template de code pour les questions et les réponses

Il y a une chose importante à savoir avant d'aller plus loin. Dans Asciidocpro, les numéros de questions sont incrémentés de un en un sans jamais être remis à zéro. Ce comportement est obligatoire car les chapitres et ou les exercices qui constituent un "livre" sont injectés après leur création. Il n'est donc pas possible de connaître l'enchaînement des questions à l'avance.

9.2.1. Le code pour poser une question

Si vous voulez poser une question, faites-le en adoptant ce template :

```
[.question] ①
**** ②
*Q{counter:_question})* ③
④
//end _question ⑤
```

**** ⑥

- ① rôle : permet d'identifier le bloc compris entre les * comme étant une question.
- ② début du bloc de la question
- ③ lettre Q pour "question" suivi du numéro de celle-ci. C'est le processeur asciidoc qui gère l'incrémentation de la valeur de l'attribut `_question.counter`: indique au processeur qu'il faut incrémenter la valeur courant de `_question`.
- ④ la question à poser qui peut contenir tout ce que vous voulez, y compris d'autres blocs de code, des "admonitions", etc
- ⑤ commentaire permettant d'identifier visuellement la fin de la question. C'est utile lorsque vous avez des blocs à l'intérieur de votre question
- ⑥ fin du bloc de la question

Comme vous l'aurez remarqué, le template de code utilise l'attribut `_question`. Cet attribut contient le numéro de question courant. Le mot `counter` permet d'incrémenter la valeur stockée dans `_question`.

Avec cette façon de faire, vous n'avez plus à vous soucier de la numérotation de vos question.

9.3. Le code pour écrire une réponse

Si vous souhaitez apporter une réponse à une question posée avec le [template de code d'une question](#), vous devez adopter le code suivant :

```
ifeval::[_show_correction] == 1] ①
[.answer] ②
**** ③
_Correction de Q{_question}_ ④
⑤

**** ⑥
//end _show_correction ⑦
endif::[] ⑧
```

- ① test qui évalue s'il faut ou non afficher la réponse grâce à la valeur de l'attribut `_show_correction`. Vous pouvez influencer le résultat de ce test en précisant la valeur de l'attribut `_user_show_correction` depuis le fichier `config/custom/attributes/user_application_attributes.adoc`
- ② rôle : permet d'identifier le bloc comme étant une réponse.
- ③ début du bloc de la réponse
- ④ référence à la question dont dépend la réponse. La valeur de l'attribut `_question` est celle de la dernière question rencontrée par le processeur asciidoc.
- ⑤ contenu de la réponse qui peut contenir tout ce que vous voulez, y compris d'autres blocs de code, des "admonitions", des images, etc

- ⑥ fin du bloc de la réponse
- ⑦ commentaire permettant de repérer visuellement la fin de la question. C'est très utile lorsqu'une réponse contient des blocs.

J'attire votre attention sur le fait que la réponse est placée dans un bloc qu'il est possible de rendre ou non dans le document final.

Le test évalue la valeur de l'attribut `_show_correction`. Lorsque la valeur de cet attribut est `1`, le document final généré contiendra les réponses. Si c'est la valeur `0`, les réponses ne seront pas affichées. C'est très utile pour générer un support sans les corrections et un support avec les réponses.

9.4. Le code pour écrire une note spéciale pour le professeur

Parfois, il est utile d'ajouter des notes qui n'ont pas vocation à apparaître dans le document distribué aux étudiants ou tout autre utilisateur du support final. Mais le professeur, le formateur ou du moins la personne qui se place comme tel peut vouloir disposer de ces notes dans son document.

Asciidocpro prévoit un template de code pour cela :

```
ifeval::[_show_note_prof] == 1] ①
.Note pour le professeur ②
[.note_prof] ③
**** ④
⑤
//end note_prof ⑥
**** ⑦
//end _show_note_prof ⑧
endif::[] ⑨
```

- ① test qui évalue s'il faut ou non afficher la note grâce à la valeur de l'attribut `_show_note_prof`. Vous pouvez influencer le résultat de ce test en précisant la valeur de l'attribut `_user_show_note_prof` depuis le fichier `config/custom/attributes/user_application_attributes.adoc`
- ② titre du bloc qui va être affiché
- ③ rôle permettant d'identifier le bloc comme étant une note pour le professeur
- ④ délimiteur du début du bloc contenant le texte de la note
- ⑤ zone dans laquelle écrire la note pour le professeur. N'importe quel contenu peut être écrit ici.
- ⑥ commentaire indiquant la fin de la note. C'est très utile lorsque la note contient d'autres blocs. Cela permet de s'y retrouver visuellement.
- ⑦ délimiteur de fin de la note
- ⑧ commentaire indiquant la fin du test. C'est utile pour s'y retrouver visuellement lorsque la note

contient d'autres tests.

⑨ fin du test d'affichage de la note

9.5. Et si je préfère utiliser mes propres templates de code ?

Vous pouvez tout à fait décider d'utiliser vos propres templates de code.

Cependant, si vous devez intégrer des éléments provenant d'autres rédacteurs ou partager les vôtres, il vous faudra procéder à une refactorisation du code pour pouvoir tout assembler dans un document final.

De plus, AsciiDocpro intègre nativement les templates de code qui vous sont proposés.

Enfin, si vous décidez d'utiliser AsciiDocproM ultérieurement, il faut adopter ces templates de code.

9.6. C'est lourd d'adopter les templates de code conseillés !

Ce peut être pénible de devoir écrire les templates recommandés.

Je vous rappelle que vous utilisez (normalement, sinon je vous plains) un IDE et que ce dernier dispose d'une fonctionnalité qui permet de créer des lives templates.

Ainsi, lorsque j'écris la lettre `q` et que je tabule, j'ai mon template de question automatiquement généré. J'écris la lettre `r` et c'est le template de réponse qui est généré.

C'est au final encore moins lourd que d'écrire le mot "question" !

10. Injecter automatiquement un template de fichier d'entête pour un "livre"

AsciiDocpro vous permet d'injecter automatiquement un template après la table des matières d'un "livre".

Si vous souhaitez injecter pour chacun de vos "livres" un contenu qui est toujours le même, vous pouvez le faire via un template d'entête.

Prenons pour exemple l'ajout d'un copyright dans chacun des "livres" généré.



Le template d'entête d'un "livre" doit être créé dans le répertoire `config/custom/templates/`.

Le nom du fichier est libre mais il doit être défini dans l'attribut `_user_header_book_template_filename` dans le fichier `config/custom/attributes/user_application_attributes.adoc` :

```
//autres attributs
// ...
:user_header_book_template_filename: book_template.adoc ①
```

① le fichier de template d'entête à utiliser pour un livre est `book_template.adoc`

Il ne faut surtout pas laisser de ligne vide même à la fin du fichier.

Maintenant, il faut créer le fichier `config/custom/templates/book_template.adoc`.

Vous êtes totalement libre dans le contenu du template.

Voici un exemple de contenu du fichier :

```
[discrete]
= Copyright

« Tous droits de reproduction, d'adaptation et de traduction, intégrale ou partielle
réservés pour tous pays. L'auteur ou l'éditeur est seul propriétaire des droits et
responsable du contenu de ce livre. »

« Le Code de la propriété intellectuelle interdit les copies ou reproductions
destinées à une utilisation collective. Toute représentation ou reproduction intégrale
ou partielle faite par quelque procédé que ce soit, sans le consentement de l'auteur
ou de ses ayant droit ou ayant cause, est illicite et constitue une contrefaçon, aux
termes des articles L.335-2 et suivants du Code de la propriété intellectuelle »

<<<

[NOTE]
====
Que vous soyez un particulier ou un auteur auto-entrepreneur, vous devez indiquer
votre nom et votre adresse. Pas votre pseudonyme.
====
```

Je n'ai utilisé que des éléments issus du langage AsciiDoc. `[discrete]` permet d'indiquer que le titre utilisé ne doit pas être repris dans la table des matières. `<<<` permet de forcer le saut de page si jamais vos titres de chapitres ne sont pas configurés pour être placés automatiquement sur une nouvelle page (comportement par défaut).

Une fois votre fichier créé, il faut indiquer que le template doit être affiché en configurant l'attribut `_user_show_header_templates` dans le fichier `config/custom/attributes/user_application_attributes.adoc` :

```
//autres attributs
// ...
:user_header_book_template_filename: book_template.adoc
:user_show_header_templates: 1 ①
```

① tous les templates d'entête définis seront affichés.

A partir de maintenant, tous vos "livres" contiendront votre template sans que vous n'ayez à faire quoi que ce soit !



Si vous ne voulez pas rendre le template d'entête d'un "livre" mais quand même les templates de chapitre et / ou d'exercice, commentez ou supprimez la ligne qui fixe le fichier à utiliser.

11. Injecter automatiquement un template de fichier d'entête pour un chapitre

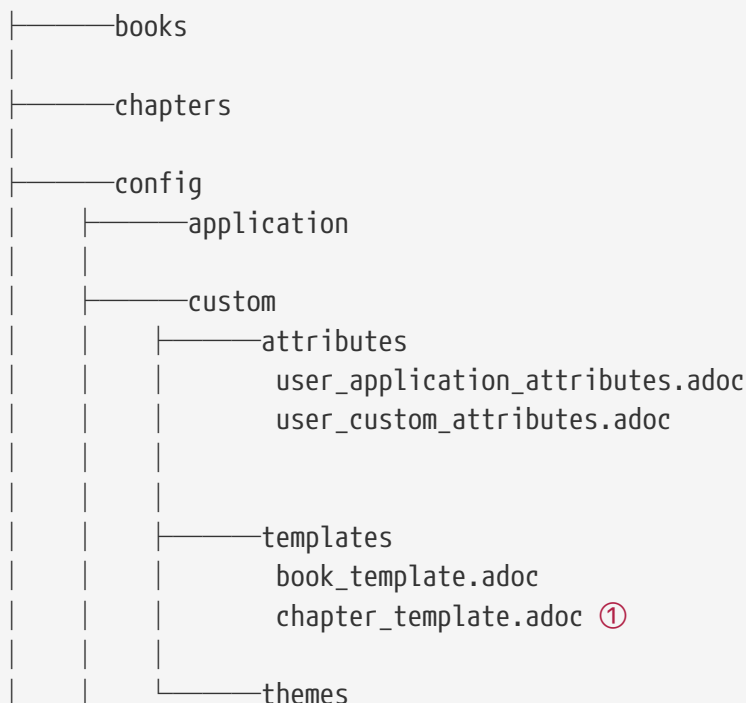
AsciiDocpro vous permet d'injecter automatiquement un template après le titre d'un chapitre.

Le principe d'injection automatique d'un template est le même que celui d'un [template d'entête de "livre"](#).

Commencez par créer un nouveau fichier asciidoc que vous placez dans le répertoire [config/custom/templates/](#). Le nom du fichier est libre.

Pour notre exemple, je crée un fichier [chapter_template.adoc](#).

Voici l'arborescence du projet



① fichier qui contient le template de chapitre. Le nom du fichier est libre.

Dans ce fichier, j'écris le contenu suivant :

[NOTE]

====

Il est attendu que vous reproduisiez tous les exemples de ce chapitre sur votre machine.

L'assimilation des notions passe par la pratique.

Toute méthode "d'étude" passive du chapitre (par exemple, lire sans se poser de question et sans chercher à réellement comprendre) ne peut être un facteur de réussite.

====

Maintenant que mon template est créé, je déclare l'attribut `_user_header_chapter_template_filename` dans le fichier `config/custom/attributes/user_application_attributes.adoc` :

```
//autres attributs
// ...
:_user_header_book_template_filename: book_template.adoc
:_user_show_header_templates: 1
:_user_header_chapter_template_filename: chapter_template.adoc ①
```

① le fichier de template d'entête à utiliser pour un chapitre est `chapter_template.adoc`.

Pour que le template que nous venons de créer soit injecté automatiquement, il faut veiller à ce que l'attribut `_user_show_header_templates` contienne la valeur `1` :

```
//autres attributs
// ...
:_user_header_book_template_filename: book_template.adoc
:_user_show_header_templates: 1 ①
:_user_header_chapter_template_filename: chapter_template.adoc
```

① l'attribut `_user_show_header_templates` était déjà défini avec la valeur 1. Cela permet d'afficher tous les templates d'entête qui seraient déclarés.

A partir de maintenant, vous verrez votre template de chapitre inséré après le titre de chaque chapitre de votre document final.

12. Injecter automatiquement un template de fichier d'entête pour un exercice

AsciiDocpro vous permet d'injecter automatiquement un template après le titre d'un exercice.

Le principe d'injection automatique d'un template est le même que celui d'un [template d'entête de "livre"](#) ou d'un [template d'entête d'un chapitre](#).

Commencez par créer un nouveau fichier asciidoc que vous placez dans le répertoire `config/custom/templates/`.

Pour notre exemple, je crée un fichier `exercise_template.adoc`. Le nom du fichier est libre.

Voici l'arborescence du projet



① fichier qui contient le template pour les exercices. Le nom du fichier est libre.

Dans ce fichier, j'écris le contenu suivant :

```

[NOTE]
====
Veillez à lire toutes les consignes.

Lorsqu'une question est constituée de plusieurs questions, faites attention à toutes
les traiter.

Il est attendu que chaque réponse soit systématiquement justifiée.

Bon courage.
====
  
```

Maintenant que mon template est créé, je déclare l'attribut `_user_header_exercise_template_filename` dans le fichier `config/custom/attributes/user_application_attributes.adoc` :

```

//autres attributs
// ...
  
```

```
:_user_header_book_template_filename: book_template.adoc
:_user_show_header_templates: 1
:_user_header_chapter_template_filename: chapter_template.adoc
:_user_header_exercise_template_filename: exercise_template.adoc ①
```

① le fichier de template d'entête à utiliser pour un chapitre est `exercise_template.adoc`.

Pour que le template que nous venons de créer soit injecté automatiquement, il faut veiller à ce que l'attribut `_user_show_header_templates` contienne la valeur `1` :

```
//autres attributs
// ...
:_user_header_book_template_filename: book_template.adoc
:_user_show_header_templates: 1 ①
:_user_header_chapter_template_filename: chapter_template.adoc
:_user_header_exercise_template_filename: exercise_template.adoc
```

① l'attribut `_user_show_header_templates` était déjà défini avec la valeur `1`. Cela permet d'afficher tous les templates d'entête qui seraient déclarés.

Veillez à ne laisser aucune ligne vide dans le fichier `config/custom/attributes/user_application_attributes.adoc` sans quoi le rendu ne sera pas généré correctement

A partir de maintenant, vous verrez votre template d'exercice inséré après le titre de chaque exercice de votre document final.

13. Partager des composants entre des chapitres de thèmes différents

Si vous avez bien lu cette documentation, il ne vous a pas échappé qu'un [chapitre doit être atomique](#).

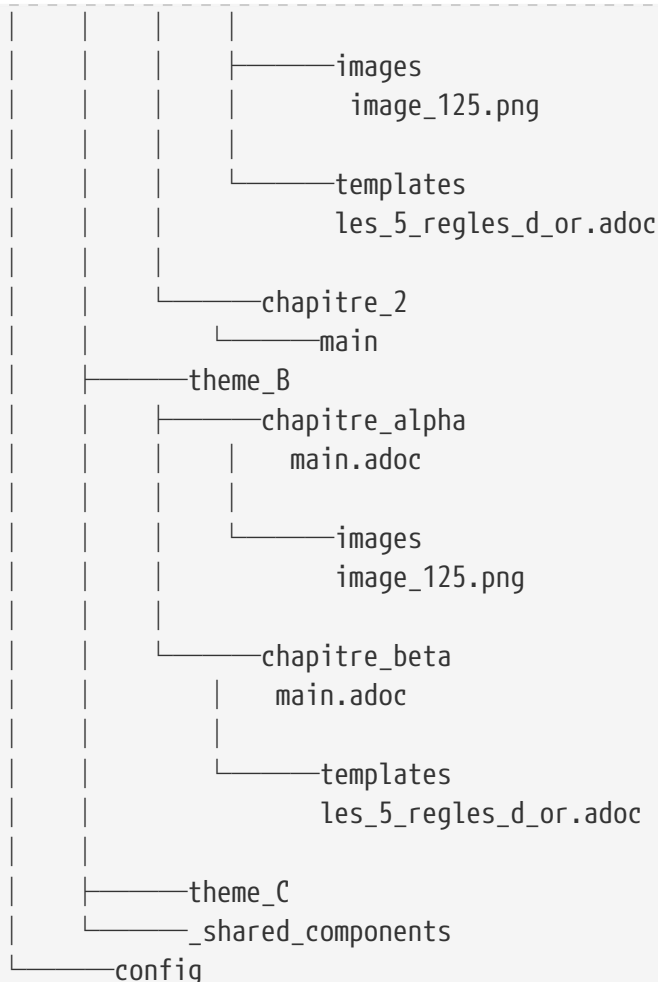
Pour cette raison, un chapitre a ses propres ressources (ses images, ses fichiers de code, etc). Ces ressources ne sont utilisables que par le chapitre qui les contient. Cela permet de déplacer un chapitre dans un autre thème sans avoir à refactoriser le contenu du chapitre.

Parfois, **plusieurs chapitres issus de thèmes différents** nécessitent une même image, un même contenu, etc.

Illustrons ce cas avec l'arborescence suivante :

```

|-----books
|-----chapters
|      |-----theme_A
|      |      |-----chapitre_1
|      |      |      |-----main.adoc
```



Le projet contient deux thèmes qui regroupent chacun deux chapitres.

Le chapitre `chapitre_1` du thème `theme_A` utilise l'image `image_125.png` et inclus 5 règles d'or qui sont dans le fichier `les_5_regles_d_or.adoc`. Le chapitre `chapitre_alpha` du thème `theme_B` utilise l'image `image_125.png` qui est la même que celle utilisée par le chapitre `chapitre_1`. Le chapitre `chapitre_beta` du thème `theme_B` inclut également le fichier `les_5_regles_d_or.adoc` qui est la même que celui utilisé par le chapitre `chapitre_1`.

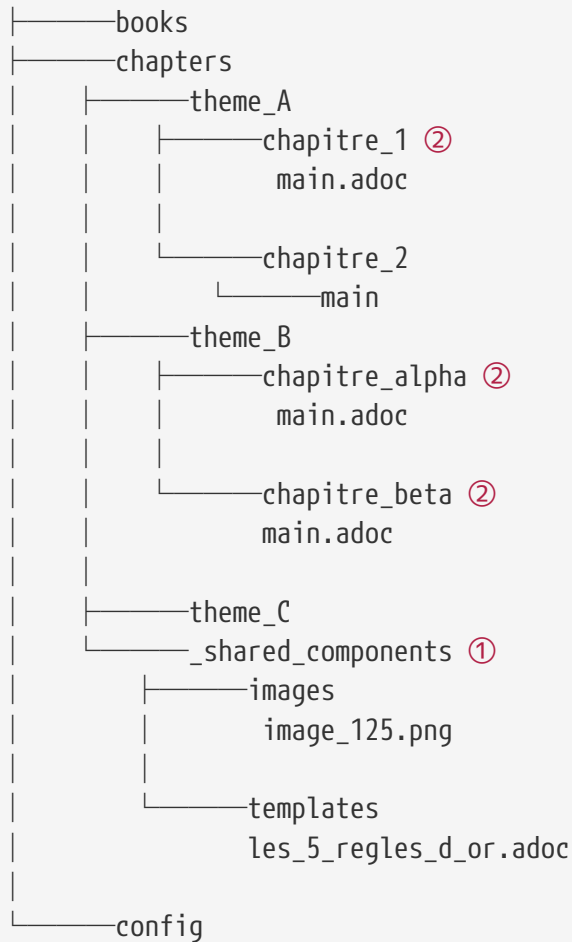
Si l'image devait être mise à jour, il faudrait faire la remplacer partout où elle est utilisée.

Si une des règles d'or venait à changer ou à être ajoutée, il faudrait mettre à jour tous les fichiers `les_5_regles_d_or.adoc`.

Si vous êtes dans ce cas, alors vous pouvez *enfreindre* la règle d'atomicité d'un chapitre. C'est-à-dire que vous pouvez "factoriser" l'image commune et le fichier des règles d'or. Ainsi, si l'image doit être mise à jour ou si le fichier des règles d'or est modifié, il n'y a qu'une seule modification à faire.

AsciiDocpro encadre cet usage de la façon suivante :

1. un dossier nommé `_shared_components` doit être créé dans le répertoire `chapters`.
2. l'image et le fichier des règles d'or sont déplacés dans le dossier `chapters/_shared_components/`.
Je conseille d'organiser le contenu de ce dossier en créant des sous-dossiers :



① Le dossier `_shared_components` créé à la racine du dossier `chapters` contient l'image partagée et les règles partagées. Les fichiers ont été rangés dans des sous-dossiers.

② Les chapitres ne contiennent plus l'image et ou le fichier des règles d'or

3. Dans le fichier `main.adoc` du chapitre `chapitre_1`, l'image et le fichier des règles sont inclus avec un chemin relatif :

```

//inclusion de l'image partagée
image:../../../../_shared_components/images/image_125.png[]
//...
//inclusion des règles d'or
include:../../../../_shared_components/templates/les_5_regles_d_or.adoc[]

```

Dans le fichier `main.adoc` du chapitre `chapitre_alpha`, l'image est incluses de la même façon :

```

//...
//inclusion de l'image partagée
image:../../../../_shared_components/images/image_125.png[]

```

Et la logique est la même pour le chapitre `chapitre_beta` qui depuis son fichier `main.adoc` inclus les règles d'or de la façon suivante :


```
//...
//inclusion des règles d'or
include:../_shared_components/templates/les_5_regles_d_or.adoc[]
```

L'avantage de cette solution est que vous pouvez déplacer un chapitre d'un thème à un autre sans rien avoir à modifier concernant les chemins des composants partagés.



L'usage d'un composant partagé doit être murement réfléchi. Il ne faudrait pas utiliser un composant partagé dans plusieurs chapitres alors que dans l'un d'eux, il n'est pas attendu d'intégrer sa mise à jour.

Exemple : vous créez un chapitre sur les 5 règles d'or à une date donnée. Vous utilisez le fichier partagé. Lorsque les règles d'or sont mises à jour, elles le sont dans le chapitre alors que celui-ci ne voulait que les règles d'or à une date donnée.

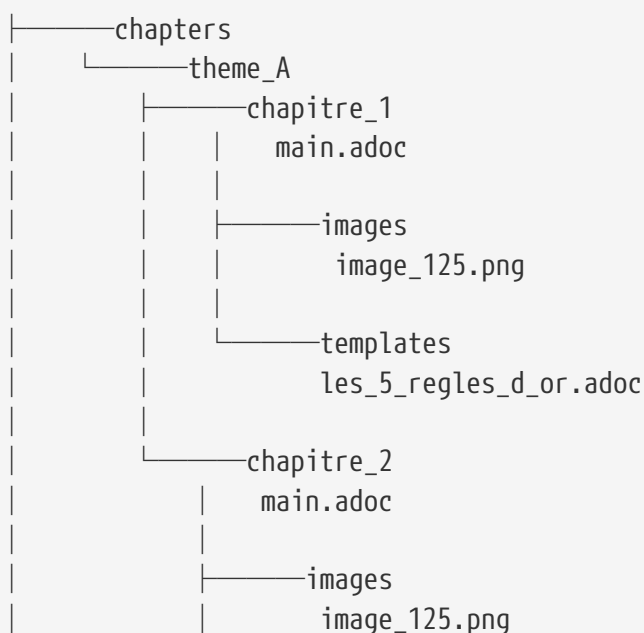
14. Partager des composants entre des chapitres d'un même thème

Si vous avez bien lu cette documentation, il ne vous a pas échappé qu'un [chapitre doit être atomique](#).

Pour cette raison, un chapitre a ses propres ressources (ses images, ses fichiers de code, etc). Ces ressources ne sont utilisables que par le chapitre qui les contient. Cela permet de déplacer un chapitre dans un autre thème sans avoir à refactoriser le contenu du chapitre.

Parfois, **plusieurs chapitres d'un même thème** nécessitent une même image, un même contenu, etc.

Illustrons ce cas avec l'arborescence suivante :

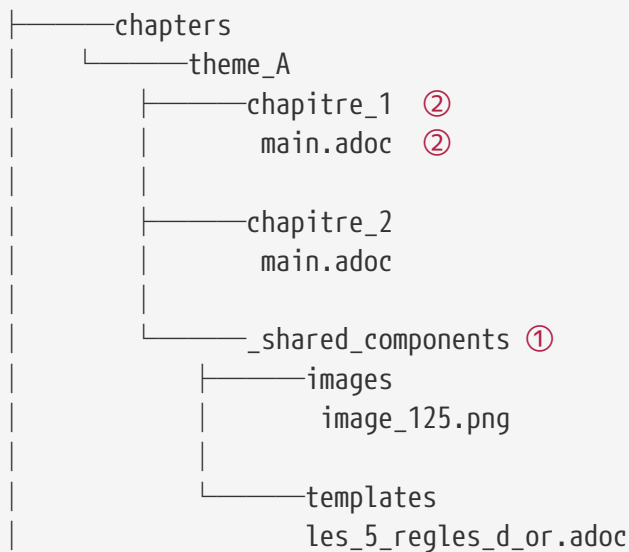




Les chapitres [chapitre_1](#) et [chapitre_2](#) utilisent la même image et le même fichier asciidoc.

Si la mise à jour de l'image est forcément répercutée dans les deux chapitres, alors il vaut mieux la partager. Cela permet de n'avoir qu'une seule mise à jour à faire. Le raisonnement est identique pour les règles d'or.

Pour faire cela, il suffit de créer un répertoire nommé [_shared_components](#) à la racine du dossier de thème dans lesquels sont placés les chapitres :



- ① Le dossier des composants partagés est placé à la racine du dossier de thème [theme_A](#) car les composants partagés ne concernent que les chapitres de ce thème. Les éléments partagés ont été rangés dans des sous-dossiers.
- ② Les chapitres n'ont plus besoin de contenir les composants partagés

Ensuite, que cela soit depuis le fichier du chapitre [chapitre_1](#) ou du chapitre [chapitre_2](#), l'inclusion de l'image et du fichier asciidoc se fait de la façon suivante :

```

//inclusion de l'image partagée
image::../_shared_components/images/image_125.png[]
//...
//inclusion des règles d'or
include::../_shared_components/templates/les_5_regles_d_or.adoc[]

```



Si un chapitre venait à être déplacé dans un autre thème, il faut avoir conscience que les liens ne seront plus valides. Il faudra alors copier les composants partagés et les placer dans le chapitre.

Pour cette raison, il faut limiter au maximum le recours aux composants partagés

entre les chapitres d'un même thème. Ne le faites que si vous êtes sûr que le chapitre restera dans son thème de départ.

15. Partager des composants entre des exercices d'un même chapitre

Si plusieurs exercices d'un même chapitre mobilisent des images communes, des templates commun, etc, vous pouvez les factoriser.

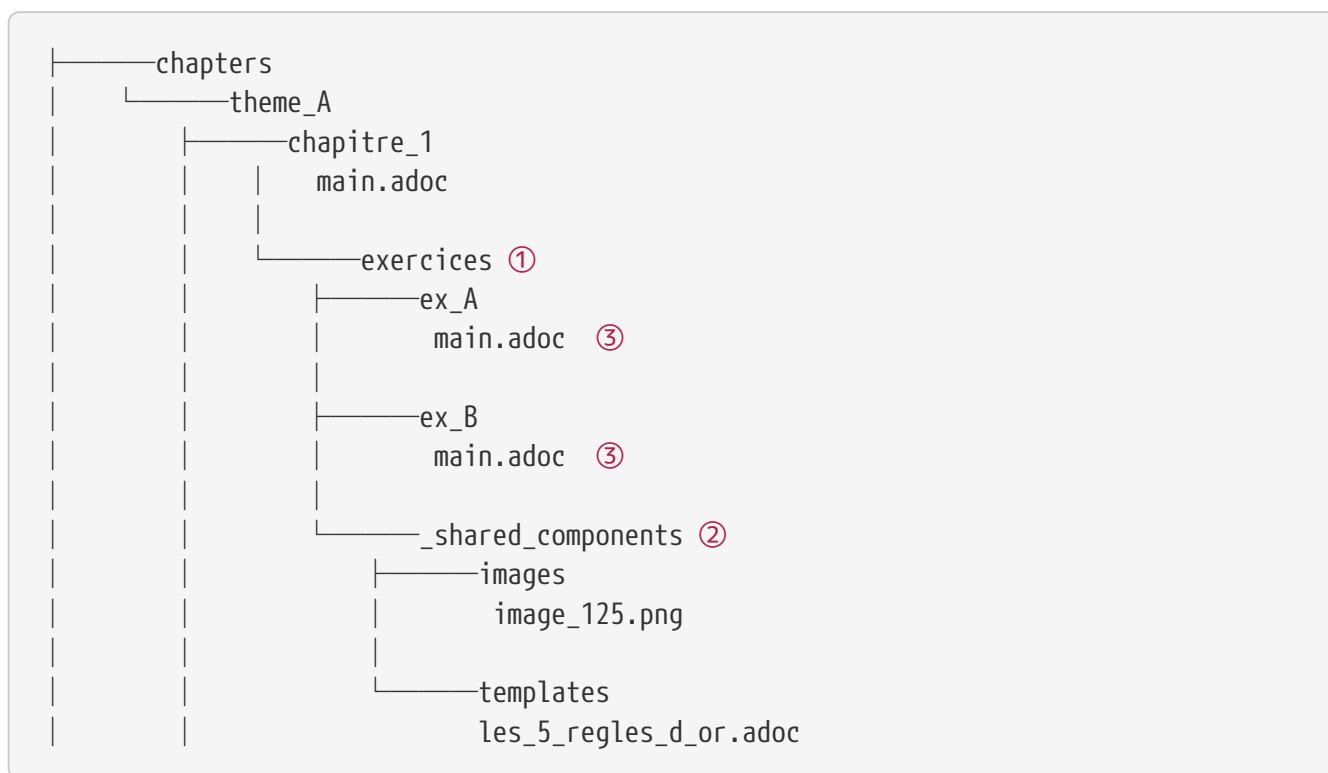
Le principe est exactement le même que pour le [partage de composants entre chapitre d'un même thème](#).

La seule différence concerne l'endroit où doit être créé le dossier `_shared_components`.

Je rappelle que chaque dossier d'exercice doit être placé dans un dossier parent nommé `exercices`.

Le dossier `_shared_components` doit être créé à la racine de celui-ci.

Voici pour exemple une arborescence qui montre le partage de composants partagés entre exercices :



- ① dossier qui contient tous les exercices du chapitre
- ② dossier des composants partagés. Ici il est organisé en sous-dossiers.
- ③ les exercices peuvent inclure les composants partagés

L'inclusion des composants partagés depuis le fichier `main.adoc` d'un exercice se fait de la façon suivante :

```
//inclusion de l'image partagée
image::../_shared_components/images/image_125.png[]
//...
//inclusion des règles d'or
include::../_shared_components/templates/les_5_regles_d_or.adoc[]
```

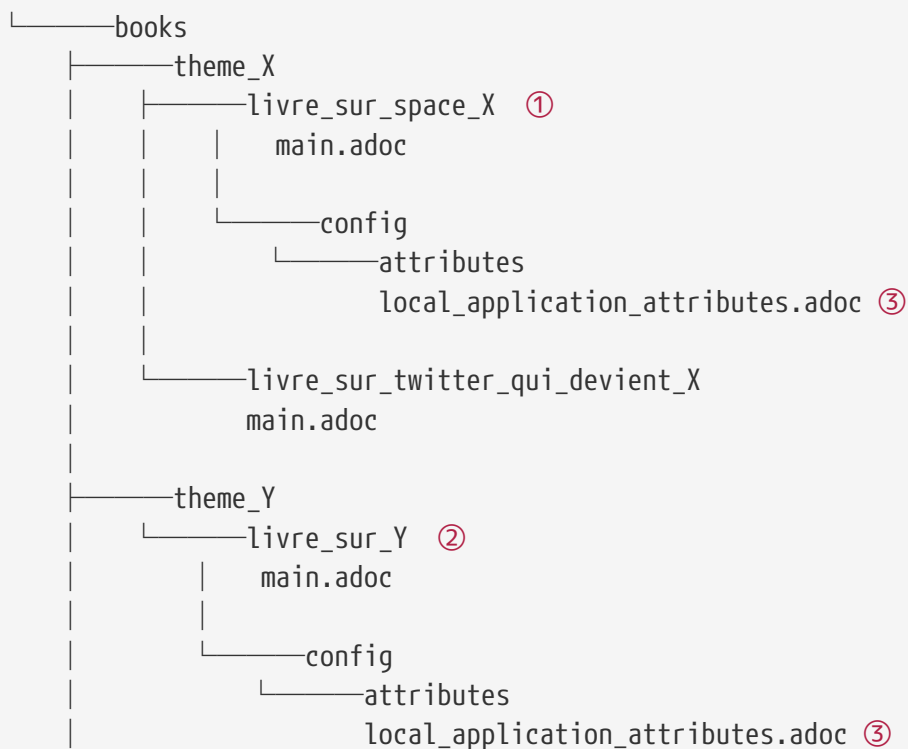
16. Partager des composants entre des livres

L'esprit des composants partagés entre des livres est différent de celui des composants partagés entre chapitres et exercices. Les éléments partagés entre des livres concernent généralement des attributs.

Si vous avez lu la partie de la documentation qui porte sur [la configuration d'attributs au niveau d'un livre](#), vous savez que vous pouvez écraser la [configuration globale](#).

Si vous êtes amené à utiliser au niveau de plusieurs livres une configuration locale qui se répète, vous pouvez opter pour les composants partagés.

Imaginons que vous ayez cette arborescence :



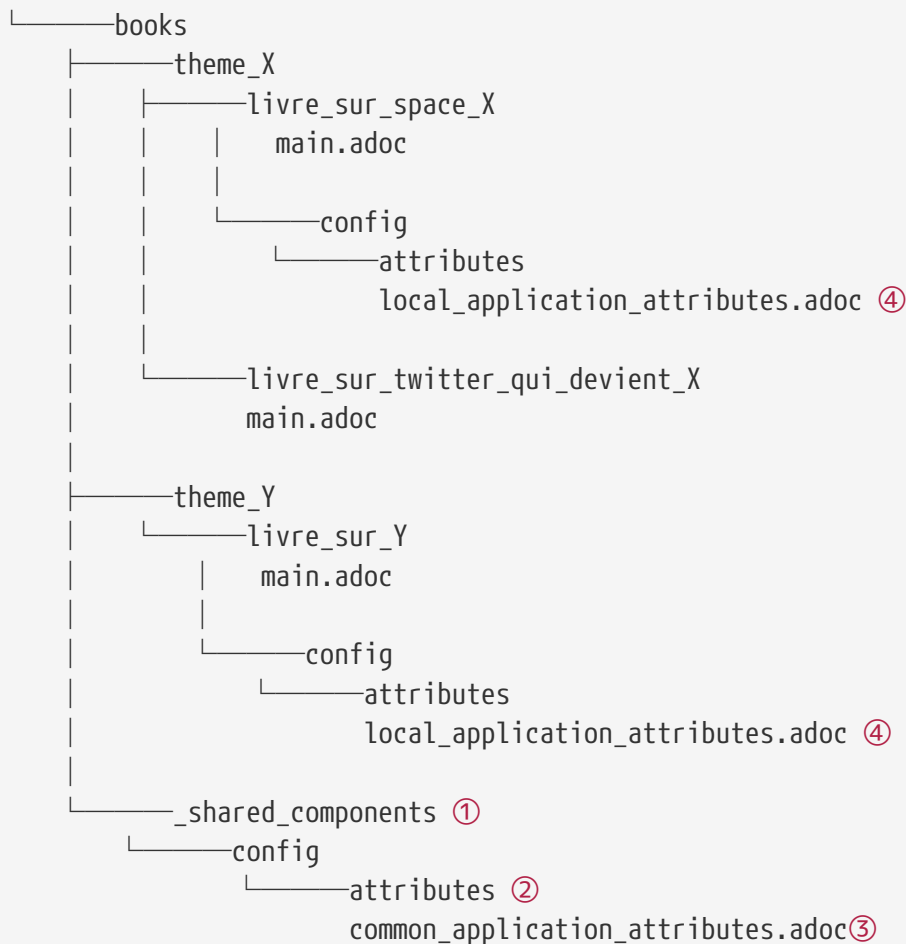
- ① Le livre utilise une configuration locale de certains attributs d'application
- ② Le livre utilise une configuration locale de certains attributs d'application identique au précédent livre
- ③ Les deux fichiers contiennent la même configuration locale :

```
:_user_show_note_prof: 0
```

```
:_user_show_correction: 0
```

Puisque les deux fichiers ont la même configuration et que cela restera toujours comme ça (sinon, il faut laisser les choses telle qu'elles sont), il peut être intéressant de factoriser les attributs dans un seul et même fichier.

Après factorisation, cela donnerait l'arborescence suivante :



- ① dossier des composants partagés
- ② j'ai repris la structure de configuration d'un livre en créant un dossier `config/attributes/` dans le dossier des composants partagés.
- ③ J'ai créé un fichier `common_application_attributes.adoc` (le nom est totalement libre) qui va contenir les attributs que je souhaite mutualiser.
- ④ les fichiers sont conservés, car ils sont nécessaires au fonctionnement d'AsciiDocpro dès lors qu'il y a une configuration locale (au niveau d'un livre).

Le fichier `common_application_attributes.adoc` contient le code mutualisé :

```
:_user_show_note_prof: 1
:_user_show_correction: 1
```

Les fichiers `local_application_attributes.adoc` sont conservés mais leur contenu est modifié de

façon à inclure le fichier `common_application_attributes.adoc`

```
//éventuellement d'autres attributs ici
//...
//les attributs mutualisés doivent être inclus. Il faut remonter jusqu'au répertoire
des composants partagés
include::../../../../_shared_components/config/attributes/common_application_attribute
s.adoc[]
```

Vous venez de voir un exemple avec peu d'attributs mais parfois, les configurations peuvent devenir plus complexes. Dans ce cas, l'utilisation de composants partagés prend tout son sens.

Voici un autre exemple qui vous montre l'intérêt d'avoir des composants à partager entre livres.

Admettons qu'au niveau global, les réponses et les notes du professeur sont affichées. Cependant, vous voulez configurer plusieurs rendus pour un même livre sans écraser la configuration globale. A ce titre, vous souhaitez pouvoir générer à partir du même livre :

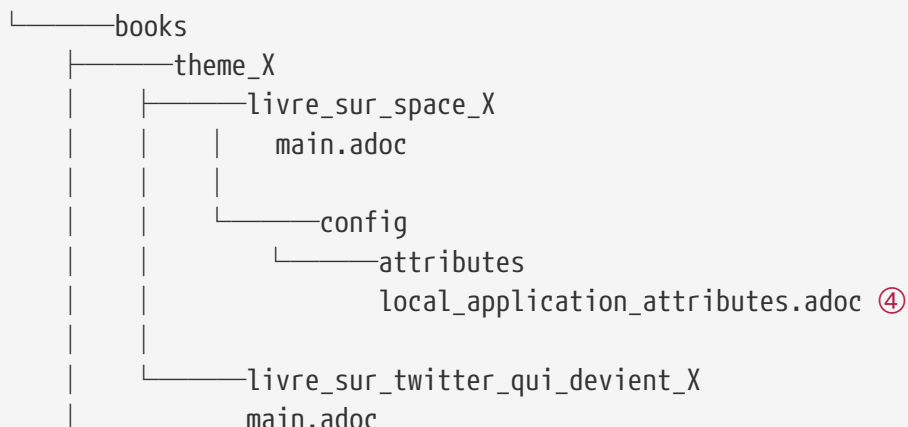
- un document sans afficher [les réponses](#) ni les [notes du professeur](#)
- un document qui affiche les réponses mais pas les notes du professeur
- un document qui affiche les réponses et les notes du professeur

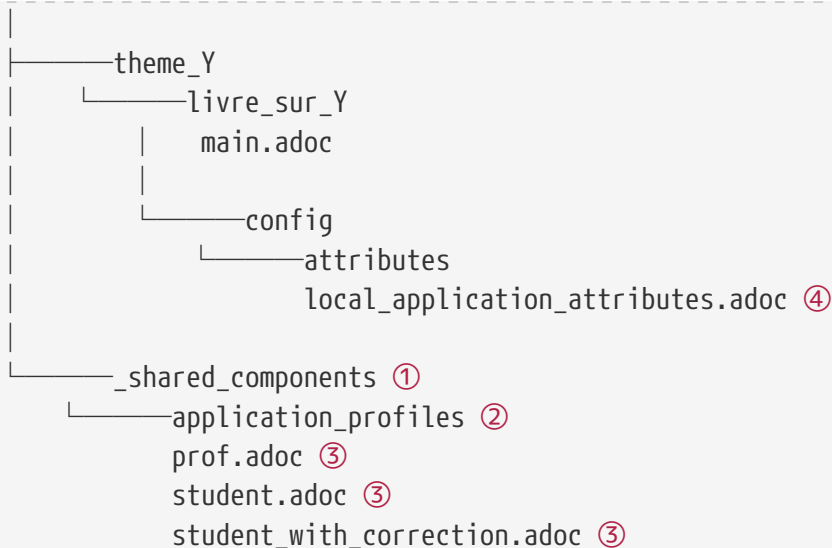
Il peut être intéressant de prévoir un "profil" par besoin.

Cela est faisable en commençant par créer un dossier nommé `_shared_components` à la racine du dossier `books`. Dans ce dossier, vous vous organisez comme vous le souhaitez. Par exemple, je crée un dossier `application_profiles` dans lequel je vais préparer trois profils :

- un profil "student" qui va configurer les attributs d'application de façon à générer un support destiné à des étudiants, c'est-à-dire sans les réponses et les notes du professeur
- un profil "student_with_correction" de façon à générer un support destiné aux étudiants mais avec la correction
- un profil "prof" qui va permettre de générer un support destiné au formateur. Ce support contiendra les réponses et les notes du professeur.

Voici l'arborescence obtenue après création des fichiers :





- ① dossier des composants partagés
- ② dossier qui contient les différents profils qui impactent le rendu final
- ③ fichier contenant des attributs d'application avec des valeurs spécifiques à un contexte recherché
- ④ fichier de configuration locale qui va inclure le fichier de profil à utiliser

Voici le code à utiliser depuis un fichier `local_application_attributes.adoc` qui doit charger un profil :

```

// éventuellement d'autres attributs
//...
// si l'on souhaite charger le profil étudiant
// include:../../../../_shared_components/application_profiles/student.adoc[]
// si l'on souhaite charger le profil étudiant avec les corrections
//
include:../../../../_shared_components/application_profiles/student_with_correction.a
doc[]
// si l'on souhaite afficher les réponses et les notes du professeur
\include:../../../../_shared_components/application_profiles/prof.adoc[] ①
  
```

- ① Pour utiliser un profil, il suffit de l'inclure depuis le fichier de configuration locale. Pour générer un fichier en fonction d'un profil voulu, il suffit de décommenter la ligne adéquate et de commenter celle qui ne l'est plus.

Avec cette possibilité, vous pouvez facilement configurer les rendus de vos livres.

17. Utilisation avec asciidoctor-diagram

Si vous utilisez `asciidoctor-diagram` pour rendre vos diagrammes dans le document final, sachez que AsciiDocpro prévoit le fonctionnement suivant :

- les images des diagrammes sont automatiquement générées dans le dossier `diagram_images` de chaque chapitre du livre qui contient des diagrammes. Ce dossier est automatiquement créé s'il

n'existe pas.

- pour accélérer la génération d'un document final déjà généré, [asciidoctor-diagram](#) utilise un cache. Asciidocpro active le cache par défaut. Le dossier de cache sera automatiquement placé dans le répertoire des images des diagrammes d'un chapitre ([diagram_images/cache](#)).

Si jamais vous constatez un problème de rendu, par exemple un diagramme qui ne se met pas à jour, vous pouvez désactiver **temporairement** le cache avec l'attribut [_user_enable_cache_diagrams](#) en lui affectant la valeur `0` dans le fichier [config/custom/attributes/user_application_attributes.adoc](#) :

```
//d'autres attributs
//...
//désactivation du cache des diagrammes
:_user_enable_cache_diagrams: 0
```

Attention à ne laisser aucune ligne vide dans le fichier [config/custom/attributes/user_application_attributes.adoc](#), y compris à la fin du fichier. Cela entraînerait un rendu incomplet

Index

A

AsciiDoc, [1](#)

attributs d'application, [18](#)

C

chapitre, [6](#)

E

exercice, [8](#)

I

index, [17](#)

L

livre, [10](#)

T

template de code, [27](#)