Asciidocpro 4.0.0

v3 | 20/03/24 à 17:05 | Auteur : Emmanuel Ravrat

Documentation

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 3
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 **5**
3.1. Où écrire un chapitre ? 5
3.2. Règles à respecter pour commencer 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 9
3.4. Publier un ou des chapitres 9
4. Ecrire un exercice **9**
4.1. Où écrire un exercice ? 9
4.2. Règles à respecter pour commencer un fichier d’exercice 10
4.2.1. Rendre les exercices atomiques 11
4.3. Organiser les répertoires d’un dossier d’exercice 11
4.4. Utiliser des images dans un exercice 12
4.5. Publier un ou des exercices 12
5. Créer un "livre" **12**
5.1. C’est quoi un livre dans Asciidocpro ? 12
5.2. Où écrire un livre ? 13
5.3. Règles à respecter pour commencer un fichier de livre 13
5.4. Les différents "livres" qu’il est possible de créer 14
5.4.1. La structure prise en exemple 14
5.4.2. Créer un livre avec plusieurs chapitres 15
5.4.3. Un livre avec des chapitres et des exercices 16
5.4.4. Un livre qui ne contient que des exercices 18
5.4.5. Un livre avec un seul chapitre 19
5.4.6. Un livre avec un seul chapitre et ses exercices 22
5.4.7. Un livre avec un seul exercice 23
5.5. Marquer un livre comme terminé 24
6. Les attributs d’application **24**
6.1. Qu’est-ce que sont les attributs d’application ? 24

6.2. Liste des attributs d'application . . . 25
6.3. Configurez les attributs d'application au niveau global . . . 30
6.4. Configurer des attributs d'application au niveau d'un livre . . . 31
7. Les attributs de métadonnées . . . 32
7.1. Qu'est-ce que sont les attributs de métadonnées ? . . . 32
7.2. Liste des attributs de métadonnées . . . 33
7.3. Où sont affichées les valeurs des attributs de métadonnées ? . . . 34
8. Les attributs de contenu . . . 34
8.1. Qu'est-ce que sont les attributs de contenu ? . . . 34
9. Adopter les conventions rédactionnelles d'Asciidocpro . . . 36
9.1. Pourquoi suivre les conventions rédactionnelles d'Asciidocpro ? . . . 36
9.2. Convention de notation d'une question . . . 37
9.3. Convention de notation d'une réponse . . . 38
9.4. Convention de notation des mots clés . . . 39
9.4.1. Marquer un mot comme étant un mot clé . . . 39
9.4.2. Créer une ancre sur un mot clé . . . 39
9.4.3. Un mot clé avec sa définition . . . 41
9.5. Convention de notation des compétences . . . 42
9.6. Convention de notation d'une note pour le professeur . . . 43
9.7. Convention de notation d'une liste de tâches à faire . . . 44
10. Utiliser des live templates . . . 45
10.1. C'est quoi un live template ? . . . 45
10.2. Des live templates dédiés à Asciidocpro . . . 46
11. Définir une image de couverture . . . 47
11.1. Utiliser une image de couverture par défaut pour tous les pdf . . . 47
11.2. Utiliser une image de couverture spécifique à un livre . . . 47
12. Définir une image de fond sur toutes les pages . . . 48
12.1. Utiliser une image de fond par défaut pour tous les pdf . . . 48
12.2. Utiliser une image de fond pour un livre spécifique . . . 48
13. Injecter automatiquement une page d'entête pour un "livre" . . . 49
14. Injecter automatiquement une page d'entête à chaque chapitre . . . 50
15. Injecter automatiquement une page d'entête à chaque exercice . . . 52
16. Partager des composants entre des chapitres de thèmes différents . . . 53
17. Partager des composants entre des chapitres d'un même thème . . . 56
18. Partager des composants entre des exercices d'un même chapitre . . . 58
19. Partager des composants entre des livres . . . 59
20. Utilisation avec asciidoctor-diagram . . . 62
21. Mise à jour d'Asciidocpro . . . 63
21.1. Comment suivre les mises à jour ? . . . 63
21.2. Mettre à jour sa version d'Asciidocpro . . . 64
21.3. Quelques astuces pour des mises à jour rapides . . . 65

21.3.1. Rechercher et remplacer une ligne dans plusieurs fichiers 65
Index **67**

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.

La documentation est en plus très bien faite : https://docs.asciidoctor.org/.

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 débuter 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` :

```
└──────chapters
    └──────un_theme_Y ①
        ├──────nom_du_chapitre_A ②
        │       main.adoc ③
        │
        └──────nom_du_chapitre_X ④
                main.adoc ①
```

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'agisse 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 retrouver rapidement un chapitre ou un exercice par son nom.

Si vous vous sentez perdu dans vos chapitres, 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 commencer un fichier de chapitre

Rappel : Le nom du fichier d'un chapitre doit être `main.adoc`.

Voici le code minimal à écrire pour débuter un fichier de chapitre :

```
:_chapter: ①
[[chapitre_Ici_le_titre_de_mom_chapitre]] ②
= Ici le titre de mom chapitre ③
include::../../../run_app.adoc[] ④
```

① **Attribut de métadonnée obligatoire** indiquant qu'il s'agit d'un chapitre

② Identifiant qui peut servir d'ancre. Cet élément est facultatif mais très utile pour faire des liens depuis un exercice ou un chapitre vers le chapitre courant.

③ Titre du chapitre qui doit être un titre de niveau 0 (soit avec un signe `=`).

④ ligne de démarrage à placer immédiatement sous le titre sans laisser de ligne vide.

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

Des attributs de métadonnées peuvent être spécifiés avant le titre du chapitre.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `start_chapter` pour générer les instructions de début du fichier `main.adoc` d'un chapitre. Le curseur est automatiquement positionné là où écrire le titre du chapitre.

Une fois ce contenu minimum spécifié, vous pouvez tranquillement dérouler le contenu de votre chapitre. Toutefois, veillez à bien délimiter le contenu du chapitre.

3.3. Bien délimiter le contenu d'un chapitre

3.3.1. Rendre les chapitres atomiques

Un chapitre est un fichier Asciidoc qui contient une division de votre support final. Il 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 pas 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 / exercice. 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.

Pour illustrer le contenu d'un chapitre, voici un dossier de chapitre :

```
└──────requete_xmlhttprequest_get ①
    │   main.adoc ②
    │
    ├──────assets ③
    │    │   questions.pdf
    │    │
    │    ├──────exemple_1
    │    │       form.html
    │    │       script.php
    │    │
    │    └──────exemple_2
    │           form.html
    │           script.php
    │
    ├──────code ④
    │    ├──────requete_xhr_get_without_parameters
    │    │       index.html
    │    │       script.php
    │    │       xhr_get_without_parameters.js
```

```
    |    |
    |    └──────requete_xhr_get_with_parameters
    |           index.html
    |           script.php
    |           xhr_get_with_parameters.js
    |
    ├──────exercises ⑤
    |    └──────soumettre_formulaire_avec_ajax ⑥
    |        |    main.adoc
    |        |
    |        ├──────assets
    |        |        form.html
    |        |        script.php
    |        |
    |        ├──────code
    |        └──────images
    ├──────extras ⑦
    |        documentation_officielle_mdn_ajax.pdf
    |
    └──────images ⑧
           inspecteur_wtih_query_string_xmlhttp.png
           liste_nombres_aleatoires.png
           page_avec_form_intevalle.png
           rendu_index_html.png
```

① dossier de chapitre

② fichier qui contient le texte du chapitre

③ dossier `assets` dont le contenu doit être communiqué, en totalité ou progressivement, aux étudiants, élèves, … car ils en ont besoin pour réaliser l'étude du chapitre. Dans le cas présent, il y a un fichier qui contient des questions et deux dossiers contenant des extraits de code qu'ils doivent lancer sur leur machine

④ dossier qui contient les extraits de code utilisés dans le chapitre. Il est important de bien organiser le contenu de chaque dossier.

⑤ dossier des exercices du chapitre (voir le chapitre sur les exercices)

⑥ dossier qui contient un exercice. Il contient lui-même des ressources, du code et des images, c'est-à-dire **la même structure que celle d'un chapitre** (voir le chapitre sur les exercices).

⑦ dossier contenant des éléments utiles au rédacteur (des notes, des annexes, des documentations, et tous les éléments utiles de prêt ou de loin à la rédaction du chapitre)

⑧ dossier des images.

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 ou dans un autre projet Asciidocpro. 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[]
```

3.4. Publier un ou des chapitres

Lorsque vous travaillez sur un chapitre, vous créez un "morceau" d'un futur "livre". Ce n'est donc pas un chapitre que vous devez publier, mais un "livre".

Vous pouvez publier :

- un livre avec plusieurs chapitres
- un livre avec un seul chapitre
- un livre avec des chapitres et leurs exercices
- un livre qui ne contient que des exercices

Tous ces points sont abordés dans le chapitre créer un livre.

4. Ecrire un exercice

4.1. Où écrire un exercice ?

Un exercice doit être lié à un chapitre. Cela signifie qu'**un exercice est créé dans un dossier de chapitre**. Le chapitre concerné peut n'avoir aucun contenu, (c'est-à-dire pas de fichier `main.adoc`), l'important étant de placer l'exercice dans un dossier de chapitre.

Par contre, il faut être vigilant au moment de créer l'exercice.

Dans le dossier de chapitre, il faut créer un sous-dossier nommé `exercises`. Ce sous-dossier va contenir tous les exercices du chapitre. Même s'il n'y a qu'un seul exercice, il faut créer ce dossier.

Ensuite, un exercice correspond à un dossier placé dans le dossier `exercises` Chaque exercice doit avoir son propre dossier **dont le nom est unique au sein du projet Asciidocpro**.

Une fois dans le dossier de l'exercice créé, créez un fichier nommé `main.adoc`. C'est dans ce fichier

que vous allez écrire le contenu de votre exercice.

Voici un exemple de chapitre qui contient deux exercices. Chaque dossier d'exercice contient un fichier `main.adoc`

```
└──────nom_du_chapitre_X ①
    │    main.adoc
    │
    └──────exercices ②
        ├──────nom_exercice_A ③
        │       main.adoc ④
        │
        └──────nom_exercice_B ③
                main.adoc ④
```

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

Une fois la structure en place, il faut suivre les règles pour commencer un exercice.

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

Rappel : Le nom du fichier d'un exercice doit être `main.adoc` et doit être placé dans un dossier spécifique à l'exercice.

Les règles à respecter pour commencer un fichier d'exercice sont les mêmes que celles d'un fichier de chapitre :

```
:_exercise: ①
[[exercice_ici_le_titre_de_lexercice]] ②
= Ici le titre de l'exercice ③
include::../../../../../run_app.adoc[] ④
```

① Attribut de métadonné obligatoire indiquant qu'il s'agit d'un exercice

② Identifiant qui peut servir d'ancre.

③ Le titre de l'exercice qui doit être un titre de niveau 0 (soit avec un signe `=`).

④ ligne de démarrage à placer immédiatement sous le titre sans laisser de ligne vide.

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

Des attributs de métadonnées peuvent être spécifiés avant le titre.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `start_exercise` pour générer les instructions de début du fichier `main.adoc` d'un exercice. Le curseur est automatiquement positionné là où écrire le titre de l'exercice.

4.2.1. Rendre les exercices atomiques

Un exercice doit suivre la même philosophie qu'un chapitre. Il doit être atomique.

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

Lorsque vous n'êtes pas sûr de devoir faire un nouveau dossier d'exercice, posez vous la question de savoir s'il faut donner un nouveau contexte pour le réaliser. Si la réponse est positive, il faut créer un autre exercice.

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 d'exercice. 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 / exercice. Le contenu de ce dossier est une ressource pour l'auteur et n'a pas vocation à être diffusé.

Voici un exemple de dossier contenant un exercice :

```
    |
    ├──────exercises ①
    |    └──────soumettre_formulaire_avec_ajax ②
    |        |    main.adoc ③
    |        |
```

```
    |       ├──────assets
    |       |       form.html
    |       |       script.php
    |       |
    |       ├──────code
    |       └──────images
```

① Ce dossier contient les exercices du chapitre

② Dossier de l'exercice. Il a la même structure qu'un chapitre.

③ fichier Asciidoc contenant le texte de l'exercice

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[]
```

4.5. Publier un ou des exercices

Lorsque vous travaillez sur un exercice, vous créez un "morceau" d'un futur "livre". Ce n'est donc pas un exercice que vous devez publier, mais un "livre".

Vous pouvez publier :

- un livre avec plusieurs chapitres
- un livre avec un seul chapitre
- un livre avec des chapitres et leurs exercices
- un livre qui ne contient que des exercices

Tous ces points sont abordés dans le chapitre créer un livre.

5. Créer un "livre"

5.1. C'est quoi un livre dans Asciidocpro ?

Un livre dans Asciidocpro est un support publiable (au format pdf, html, etc.) composés d'un seul ou de plusieurs chapitres, avec ou sans leurs exercices ou composé exclusivement d'un ou de plusieurs exercices.

C'est là que le fait d'avoir écrit des chapitres atomiques et des exercices atomiques va vous être très

utile. Grâce à cette atomicité, il est facile de déterminer les différents chapitres et / ou exercices qui doivent composer un "livre".

5.2. Où écrire un livre ?

Tout ce qui concerne les "livres" se passe dans le dossier `books` du projet Asciidocpro.

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, c'est-à-dire comme un dossier qui regroupe des "livres" autour d'un même sujet. **Le nom de ce dossier doit être unique dans tout le projet Asciidocpro**.

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

```
└──────books ①
    └──────theme_Z ②
        ├──────nom_du_livre_4 ③
        │       main.adoc ④
        │
        └──────nom_du_livre_B ③
                main.adoc ④
```

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 référence les parties du "livre"

5.3. Règles à respecter pour commencer un fichier de livre

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

Rappel : chaque "livre" correspond à un dossier lui-même créé dans un sous dossier du dossier `books` (voir où écrire un livre).

Dans le dossier du "livre", créez un fichier `main.adoc`.

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 :

```
:_book: ①
[[book_ici_le_titre_de_mom_chapitre]] ②
= Ici le titre du livre ③
include::../../../run_app.adoc[] ④
```

① Attribut obligatoire indiquant qu'il s'agit d'un chapitre

② Identifiant qui peut servir d'ancre. Cet élément est facultatif mais pourrait s'avérer utile si à l'avenir, il devenait possible de faire un "livre" de "livres".

③ Le titre du "livre" qui doit être un titre de niveau 0 (soit avec un signe `=`).

④ ligne de démarrage à placer immédiatement sous le titre sans laisser de ligne vide.

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

Des attributs de métadonnées peuvent être spécifiés avant le titre du livre.

Le livre peut être personnalisé en définissant des attributs d'application au niveau du livre

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `start_book` pour générer les instructions de début du fichier `main.adoc` d'un livre. Le curseur est automatiquement positionné là où écrire le titre du livre.

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

Il est possible de créer :

- un support contenant plusieurs chapitres avec ou sans exercices
- un support contenant un seul chapitre avec ses exercices
- un support ne contenant que des exercices
- un support contenant un seul chapitre
- un support contenant un seul exercice

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 :

```
├──────books
│     └──────theme_Z
│          ├──────nom_du_livre_A
│          │        main.adoc
│          │
│          └──────nom_du_livre_B
│                  main.adoc
│
└──────chapters
     └──────mon_theme
```

```
        ├──────nom_du_chapitre_A
        │     │    main.adoc
        │     │
        │     └──────exercices
        │           └──────nom_exercice_F
        │                  main.adoc
        │
        └──────nom_du_chapitre_X
              │    main.adoc
              │
              └──────exercices
                    ├──────nom_exercice_A
                    │       main.adoc
                    │
                    └──────nom_exercice_B
                           main.adoc
```

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` :

```
:_book:
[[book_Asciidocpro]]
= Asciidocpro
include::../../../run_app.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` :

```
:_book:
[[book_Asciidocpro]]
= Asciidocpro
include::../../../run_app.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ①
include::{_add_chapter}[] ②
```

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

② inclusion du fichier qui permet d'injecter le chapitre précisé dans l'attribut `_chapter_folder_path`.

Pour ajouter un second chapitre, il faut procéder de la même façon. Ajoutons le chapitre `nom_exercice_B` après le chapitre précédemment inséré.

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

```
:_book:
[[book_Asciidocpro]]
= Asciidocpro
include::../../../run_app.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X //
include::{_add_chapter}[]

//ajout d'un second chapitre
:_chapter_folder_path: chapters/mon_theme/nom_exercice_B ① ②
include::{_add_chapter}[] 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 partie du même thème que le chapitre précédemment inséré.

③ l'inclusion du fichier qui permet d'insérer le chapitre dans le "livre".

Vous pouvez ajouter autant de chapitres que vous le souhaitez.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `add_chapter` pour générer les instructions permettant d'ajouter un chapitre depuis un livre. Le curseur est automatiquement positionné là où écrire le chemin du dossier de chapitre.

Une fois satisfait, vous pouvez générer le fichier soit avec la commande d'asciidoctor-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 qui montre un "livre" qui va être constitué des éléments suivants :

- du chapitre X
- de l'exercice B du chapitre X
- de l'exercice A du chapitre X
- du chapitre A
- de l'exercice F du chapitre A

```
[[book_Asciidocpro]]
= Asciidocpro
include::../../../run_app.adoc[]

//ajout d'un premier chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ①
include::{_add_chapter}[]  ②

//ajout du premier exercice du chapitre
:_exercise_folder_name: nom_exercice_B ③
include::{_add_exercise}[] ④

//ajout du second exercice du chapitre //
:_exercise_folder_name: nom_exercice_A ⑤
include::{_add_exercise}[] ⑥

//ajout d'un second chapitre
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_A ⑦
include::{_add_chapter}[]  ⑧

//ajout de l'exercice dans le chapitre précédemment inséré
:_exercise_folder_name: nom_exercice_F ⑨
include::{_add_exercise}[] ⑩
```

① mise en mémoire du chapitre à ajouter à l'attribut `_chapter_folder_path`

② le chapitre stocké dans `_chapter_folder_path` est injecté dans le livre

③ mise en mémoire du dossier d'exercice à injecter. Ce dossier est automatiquement recherché dans le chapitre en mémoire

④ l'exercice stocké dans `_exercise_folder_name` est injecté dans le livre

⑤ mise en mémoire d'un autre dossier d'exercice. La valeur stockée dans `_exercise_folder_name` est écrasée par la nouvelle. Le dossier d'exercice est recherché dans le chapitre stocké dans `_chapter_folder_path`.

⑥ l'exercice stocké dans `_exercise_folder_name` est injecté dans le livre

⑦ mise en mémoire du nouveau chapitre à injecter. La valeur de l'attribut `_chapter_folder_path` est

écrasée par la nouvelle.

⑧ le chapitre stocké dans `_chapter_folder_path` est injecté dans le livre

⑨ mise en mémoire d'un autre dossier d'exercice. La valeur stockée dans `_exercise_folder_name` est écrasée par la nouvelle. Le dossier d'exercice est recherché dans le chapitre stocké dans `_chapter_folder_path`.

⑩ l'exercice stocké dans `_exercise_folder_name` est injecté dans le livre

Pour faire simple, si vous voulez injecter un chapitre, vous spécifiez le chemin depuis le dossier `chapters` jusqu'au nom de dossier du chapitre dans l'attribut `_chapter_folder_path`. Ensuite, vous injectez le chapitre avec `include::{_add_chapter}[]`. Pour ajouter un exercice, il faut déjà avoir ajouté le chapitre car un exercice est recherché dans le dossier précisé dans `_chapter_folder_path`. Ensuite, il suffit de préciser le nom du dossier de l'exercice dans `_exercise_folder_path` et de l'injecter avec `include::{_add_exercise}[]`.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, pour ajouter un exercice, vous pouvez utiliser le live template `add_exercise` pour générer les instructions permettant d'ajouter un exercice depuis un livre. Le curseur est automatiquement positionné là où écrire le chemin du dossier d'exercice.

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 faut commencer par créer le début du fichier "livre" `main.adoc` conformément aux règles applicables pour débuter un livre.

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

```
:_book:
[[book_mon_livre_dexercices]]
= Mon livre d'exercices
include::../../../run_app.adoc[]
```

Une fois la base en place, indiquez le chemin du chapitre depuis lequel proviennent les exercices que vous souhaitez ajouter avec l'attribut `_chapter_folder_path`. Enfin, précisez le nom de chaque dossier d'exercice de ce chapitre avec l'attribut `_exercise_folder_name` suivi de la ligne d'ajout de l'exercice :

```
:_book:
[[book_mon_livre_dexercices]]
= Mon livre d'exercices
include::../../../run_app.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
:_exercise_folder_name: nom_exercice_B ②
include::{_add_exercise}[] ③

//ajout du second exercice du chapitre
:_exercise_folder_name: nom_exercice_A ④
include::{_add_exercise}[] 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::{_add_exercise}[] ⑧
```

① mise en mémoire du chemin du dossier de chapitre depuis lequel proviennent les exercices à inclure

② mise en mémoire du nom de dossier d'exercice à injecter. Ce dossier sera recherché dans le dossier de chapitre mise en mémoire dans le dernier `_chapter_folder_path`.

③ injection de l'exercice mise en mémoire dans le livre

④ mise en mémoire du nom du dossier du second exercice à injecter. La valeur stockée dans `_exercise_folder_name` est écrasée par la nouvelle. Ce dossier sera recherché dans le dernier chapitre mémorisé dans `_chapter_folder_path`.

⑤ injection de l'exercice mise en mémoire dans le livre

⑥ mise en mémoire d'un autre dossier de chapitre. La nouvelle valeur écrase la valeur précédemment stockée dans `_chapter_folder_path`.

⑦ mise en mémoire du nom de dossier d'exercice à injecter. Ce dossier sera recherché dans le dernier chapitre mémorisé dans `_chapter_folder_path`.

⑧ injection de l'exercice mise en mémoire dans le livre

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 débuter un livre.

Si vous devez créer un "livre" qui ne contient qu'un seul chapitre, vous devez créer dans votre dossier de livre 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 avec `include::{_add_chapter}[]`.

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!
:_book: lonely ①
include::../../../run_app.adoc[]
:_chapter_folder_path: chapters/mon_sujet/nom_du_chapitre_A
include::{_add_chapter}[]
```

① L'attribut de métadonnée `_book` reçoit la valeur `lonely` pour indiquer que le chapitre est seul dans le livre.

Il ne doit pas y avoir de lignes vides avant l'injection du chapitre sans quoi le rendu ne sera pas généré correctement.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `start_book_lonely` pour générer les instructions de début du fichier `main.adoc` d'un livre ne contenant qu'un seul chapitre ou exercice. Le curseur est automatiquement positionné là où écrire le chemin du chapitre à utiliser.

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

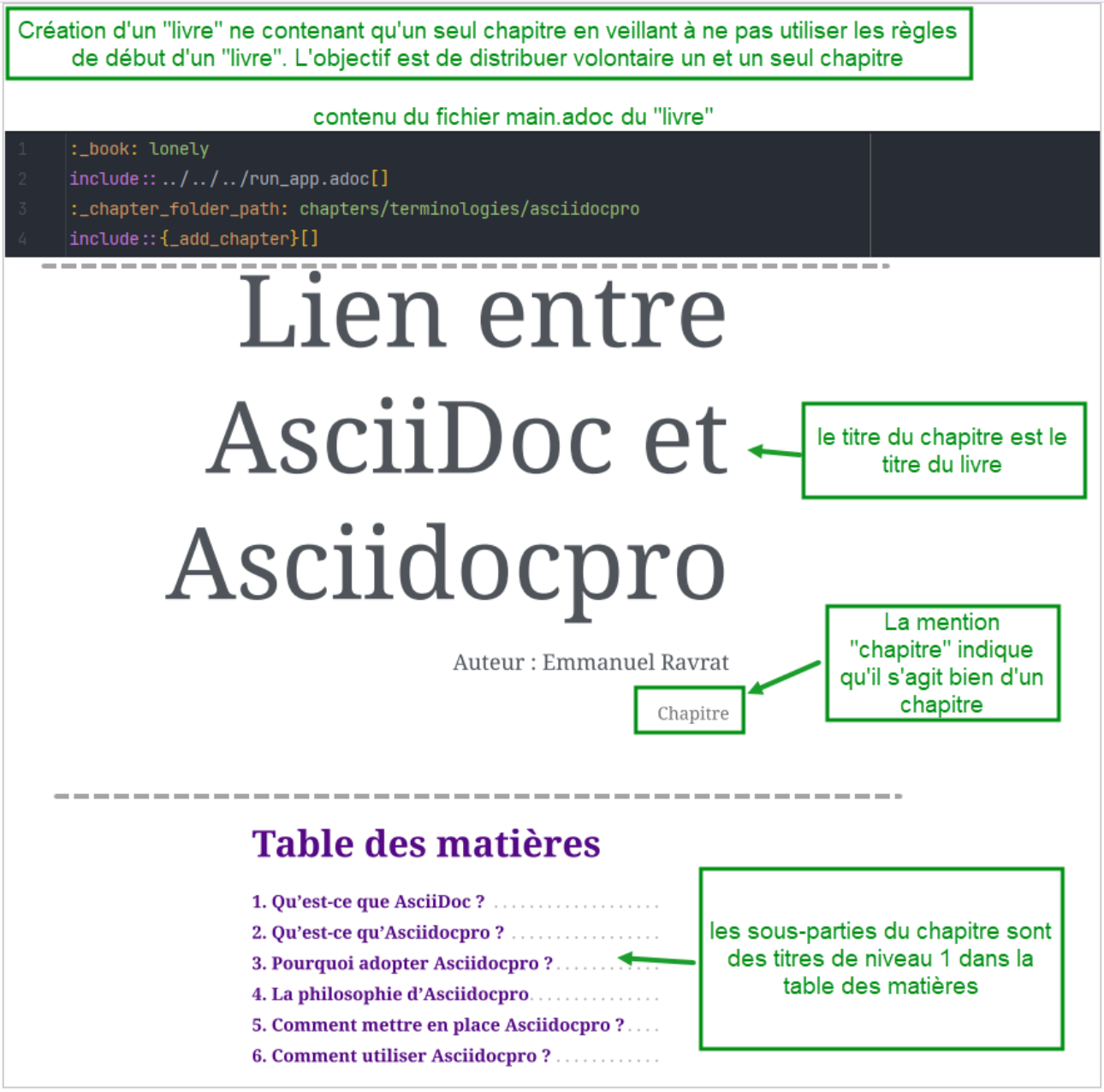
Si vous souhaitez faire évoluer votre support en ajoutant des chapitres, il vous faudra rajouter les règles à respecter pour débuter un livre.

Que se passe-t-il si vous appliquez quand même les règles à respecter pour commencer un livre ?

Une image vaut mille explications paraît-il. Alors voici en images le résultat de la création d'un livre dont l'objectif est de diffuser arbitrairement un et un seul chapitre.

- **Exemple 1** : le fichier `main.adoc` du dossier de "livre" mobilise les règles à respecter pour commencer un livre alors qu'il ne le devrait pas puisqu'il n'y aura qu'un seul chapitre.

- **Exemple 2** : le fichier `main.adoc` du dossier "livre" ignore les règles applicables pour commencer un livre car il ne contiendra qu'un et un seul chapitre.



Comme vous pouvez le voir sur les images, le rendu est différent.

5.4.6. Un livre avec un seul chapitre et ses exercices

Ce type de livre est particulier car il n'est pas concerné par les règles à respecter pour débuter un livre.

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 commencer un livre et partir d'un fichier `main.adoc` totalement vide depuis le dossier du "livre".

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 deux exercices :

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

```
:_book: lonely ①
include::../../../run_app.adoc[]
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ②
include::{_add_chapter}[] ③

:_exercise_folder_name: nom_exercice_B ④
include::{_add_exercise}[] ⑤

//ajout du second exercice du chapitre ⑥
:_exercise_folder_name: nom_exercice_A
include::{_add_exercise}[]
```

① L'attribut de métadonnée `_book` reçoit la valeur `lonely` pour indiquer que le chapitre sera le seul du livre.

② Le chemin du chapitre est spécifié via l'attribut `_chapter_folder_path`

③ le chapitre ciblé est injecté dans le "livre". L'ajout d'un exercice se fait après l'ajout d'un chapitre

④ 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.

⑤ inclusion du fichier qui permet d'automatiser la liaison de l'exercice au chapitre précédemment ajouté

⑥ Ajout d'un second exercice avec la même logique : le nom du dossier d'exercice est précisé dans l'attribut `_exercise_folder_name` puis la ligne `include::{_add_exercise}[]` injecte l'exercice dans le livre.

Aucune ligne vide ne doit être laissée avant l'ajout du chapitre.

5.4.7. Un livre avec un seul exercice

Ce type de livre est particulier car il n'est pas concerné par les règles à respecter pour débuter un livre.

Pour créer un livre avec un seul exercice, il faut appliquer la même démarche qu'un livre avec un seul chapitre.

Il faut écrire le bloc suivant :

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

```
:_book: lonely ①
include::../../../run_app.adoc[]
:_chapter_folder_path: chapters/mon_theme/nom_du_chapitre_X ②
:_exercise_folder_name: nom_exercice_B ③
include::{_add_exercise}[] ④
```

① L'attribut de métadonnée `_book` reçoit la valeur `lonely` pour indiquer qu'un seul chapitre est utilisé dans le livre.

② Le chemin du dossier du chapitre est spécifié via l'attribut `_chapter_folder_path` **mais il n'est pas injecté**. 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 spécifié.

Il faut veiller à ce qu'il n'y ait aucune ligne vide avant l'injection de l'exercice (ligne `include::{_add_exercise}[]`)

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 :

```
//fichier main.adoc du livre

// ... ajout de chapitres et / ou d'exercices

include::{_end_book}[] ①
```

① Ligne à écrire pour injecter l'index

Si vous ommettez cette ligne, cela ne gêne en rien la génération du document final. Cependant, vous ne verrez pas l'index dans votre livre.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `end_book` pour générer la ligne de fin d'un livre.

6. Les attributs d'application

6.1. Qu'est-ce que sont les attributs d'application ?

Les attributs sont des noms auxquels il est possible d'associer une valeur. En utilisant le nom, vous accédez à la valeur associée.

Les attributs d'application sont des noms qui permettent d'associer des valeurs qui paramètrent le comportement de l'application Asciidocpro. Ils n'existent pas dans le langage Asciidoc et ont été créés pour servir d'interface avec les attributs de ce langage. Effectivement, ils ne sont parfois pas simples à utiliser. Asciidocpro vous offre ainsi une couche d'abstraction qui rend les choses plus

faciles (paramétrer le rendu pdf, afficher le rendu dans l'IDE, modifier la mise en forme, etc.)

Tous les attributs d'application Asciidocpro sont reconnaissables car ils débutent tous par `user`.

Exemples :

- `_user_default_author` qui permet de définir un nom d'auteur au niveau de l'application
- `_user_exercise_title` qui permet de préciser le titre de la partie des exercices
- `_user_show_toc` qui permet d'afficher ou non la table des matières
- `_user_cover_image_filename` qui permet de définir le nom de l'image à utiliser sur la couverture du support pdf
- etc.

6.2. Liste des attributs d'application

Avant d'aborder l'utilisation des attributs d'application, voici la liste complète de tous les attributs que vous pouvez personnaliser avec ce qu'ils permettent de configurer et leur valeur par défaut :

```
//
// ------------------------------------------------------------------------
// Image de fond pour toutes les pages (sauf la couverture).
//
// Valeur : nom de l'image avec son extension. L'image doit être placée dans le
//dossier "images" du dossier du livre concerné
//
// Aucune valeur par défaut.
// ------------------------------------------------------------------------
:_user_background_image_for_all_pages:
//
// ------------------------------------------------------------------------
// Préfix automatique avant chaque titre de chapitre
//
// A laisser vide pour qu'il n'y ait aucun préfixe.
// ------------------------------------------------------------------------
:_user_chapter_label:
//
// ------------------------------------------------------------------------
// Image de fond de la page de couverture.
//
// L'image est centrée sur la page de couverture
//
// Valeur : nom de l'image avec son extension. L'image doit être placée dans le
//dossier "images" du dossier du livre concerné
//
// Aucune valeur par défaut.
// ------------------------------------------------------------------------
:_user_cover_image_filename:
//
```

```
// ------------------------------------------------------------------------
// 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 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 métadonnées sous les titres des chapitres et des exercices
//
// 1 pour afficher les métadonnées, 0 pour les masquer
//
// Valeur par défaut : 1
// ------------------------------------------------------------------------------
:_user_show_metadata: 1
```

```
127 //
128 // ------------------------------------------------------------------------
129 // Affichage des notes du professeur (à la condition d'utiliser le pattern de la
    documentation)
130 //
131 // 1 pour afficher, 0 pour les masquer
132 //
133 // Valeur par défaut : 1
134 // ------------------------------------------------------------------------
135 :_user_show_note_prof: 1
136 //
137 // ------------------------------------------------------------------------
138 // Affichage de la numérotation des titres
139 //
140 // 1 pour afficher, 0 pour masquer
141 //
142 // Valeur par défaut : 1
143 // ------------------------------------------------------------------------
144 :_user_show_title_numbers: 1
145 //
146 // ------------------------------------------------------------------------
147 // Afficher la table des matières (table of content)
148 //
149 // 1 pour afficher, 0 pour masquer
150 //
151 // Valeur par défaut : 1
152 // ------------------------------------------------------------------------
153 :_user_show_toc: 1
154 //
155 // ------------------------------------------------------------------------
156 // Afficher les tâches à faire
157 //
158 // 1 pour afficher, 0 pour masquer
159 //
160 // Valeur par défaut : 0
161 // ------------------------------------------------------------------------
162 :_user_show_todo: 0
163 //
164 // ------------------------------------------------------------------------
165 // Afficher les tâches à faire
166 // ------------------------------------------------------------------------
167 //
168 // ------------------------------------------------------------------------
169 // Numérotation des titres jusqu'au niveau N
170 //
171 // Valeur utilisable : de 1 à 5 (ex avec la valeur 2 : les titres de section des
    niveaux
172 //3 à 5 ne sont pas numérotés)
173 //
174 // Valeur par défaut : 5 (tous les niveaux de titre sont numérotés)
175 // ------------------------------------------------------------------------
```

```
:_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
//
// ------------------------------------------------------------------------
// Nom du type de support affiché sur la page de couverture
//
// Valeurs automatiquement générées : Livre / Chapitre / Exercice
//
// Valeur par défaut : auto (pour profiter des valeurs automatiquement générées)
// Si aucune valeur n'est précisée, le type de support ne sera pas affiché
//
// ------------------------------------------------------------------------
:_user_type_support: auto
```

Ces attributs peuvent être définis :

- au niveau global, c'est-à-dire que leur valeur sera applicable dans tous les livres, chapitres et exercices
- au niveau local (celui d'un livre). C'est-à-dire que vous pouvez redéfinir la valeur d'un attribut d'application de façon à ce qu'il ne concerne qu'un livre. Ainsi, la valeur globale est écrasée par la valeur localement définie. Cela est très pratique pour déterminer le comportement de rendu de chaque livre tout en pouvant travailler sur les chapitres et exercices avec la configuration globale.

Par exemple, lorsque vous réalisez un exercice, vous créez des questions et des réponses. Durant la rédaction, vous apprécierez de voir les réponses mais lorsque le livre doit être généré, vous souhaiterez peut être ne pas les afficher. Cela est justement rendu possible grâce à la redéfinition locale des attributs d'application.

6.3. Configurez les attributs d'application au niveau global

Il faut comprendre par "niveau global" que les attributs d'applications sont repris dans tous les livres, chapitres et exercices par défaut.

Asciidocpro utilise tous les attributs d'application avec une valeur par défaut.

Si vous souhaitez modifier l'un de ces attributs, vous devez le faire dans le fichier `config/attributes/global_application_attributes.adoc`. Si ce fichier n'existe pas dans le chemin spécifié, alors il faut le créer.

Comme son nom l'indique, ce fichier ne doit contenir que des attributs d'application (pour garder les choses bien organisées).

Le fichier `config/attributes/global_application_attributes.adoc` ne doit jamais contenir de ligne vide, y compris à la fin du fichier. Le rendu ne serait pas celui attendu

Pour comprendre comment utiliser les attributs d'application au niveau global, nous allons partir du principe que vous souhaitez la configuration globale suivante :

- la table des matières doit avoir pour titre "Sommaire"
- l'auteur par défaut sera "Chuck Norris"
- les notes du professeur ne seront pas affichées
- le fichier de thème à utiliser est `mon-super-theme.yml`

J'identifie dans la liste des attributs d'application les attributs concernés et je définis leur valeur au niveau global :

```
//fichier {_file_path_attributes_application_global_user}
//
//titre de la table des matières
:_user_toc_title: Sommaire
//
//auteur par défaut
:_user_default_author: Chuck Norris
//
//masquer les notes du professeur
:_user_show_note_prof: 0
//
//fichier de thème à utiliser
:_user_pdf_theme_filename:
```

Vous pouvez utiliser des commentaires dans le fichier si cela vous aide.

Je rappelle qu'il ne doit y avoir aucune ligne vide dans ce fichier y compris à la fin

Tous les livres, chapitres et exercices seront paramétrés avec ces valeurs dans l'IDE.

S'il faut un comportement spécifique au niveau d'un livre (ce sera souvent le cas), vous pouvez configurer les attributs d'application au niveau d'un livre.

6.4. Configurer des attributs d'application au niveau d'un livre

Lorsque vous souhaitez paramétrer les attributs d'applications avec des valeurs spécifiques pour un livre, vous pouvez **définir les attributs d'application au niveau du dossier du livre**.

Partons de l'exemple ci-dessous. Le dossier `books` contient un dossier qui regroupe deux livres sur Chuck Norris. L'objectif est de configurer des attributs d'application dans le livre "enfance_de_chuck_norris".

```
books
 |
 ├──────livres_sur_chuck_norris
 |     ├──────chuck_norris_vs_bruce_lee
 |     |        main.adoc
 |     |
 |     └──────enfance_de_chuck_norris
 |              main.adoc
```

Pour spécifier des attributs d'application propres au livre, il faut suivre les étapes suivantes :

1. dans le dossier du livre en question, créer un dossier `attributes`. Dans ce dossier, créer un fichier `local_application_attributes.adoc`. Cela donne l'arborescence suivante :

```
books
 |
 ├──────livres_sur_chuck_norris
 |     ├──────chuck_norris_vs_bruce_lee
 |     |        main.adoc
 |     |
 |     └──────enfance_de_chuck_norris
 |          |    main.adoc
 |          |
 |          └──────attributes ①
 |                   local_application_attributes.adoc ②
```

① Dossier qui contient les fichiers d'attributs définis au niveau local

② Fichier dans lequel définir les attributs d'application au niveau du livre

2. Une fois le livre créé, il faut déterminer les attributs d'application à utiliser au niveau du livre

(voir la liste complète) et les définir dans le fichier `local_application_attributes.adoc` précédemment créé. C'est en fait comme configurer des attributs d'application au niveau global à la différence qu'ils sont définis dans le fichier `attributes/local_application_attributes.adoc` du dossier du livre

3. Pour indiquer à Asciidocpro qu'il existe des attributs d'applications définis au niveau local, il faut ajouter l'attribut de métadonnée `_use_local_application_attributes` dans le fichier `main.adoc` du livre.

```
//fichier main.adoc du dossier "enfance_de_chuck_norris"
:_use_local_application_attributes: ①
:_book:
[[book_lenfance_de_chuck_norris_il_etait_deja_tres_fort]]
= L'enfance de Chuck Norris (il était déjà très fort )
include::../../../run_app.adoc[]
```

① L'attribut `_use_local_application_attributes` indique à Asciidocpro que des attributs d'application sont définis au niveau du livre dans le dossier `attributes/local_application_attributes.adoc`

Pour ne plus appliquer les attributs locaux, il suffit de commenter la ligne (il est également possible de supprimer la ligne et de supprimer le dossier des attributs locaux).

Si l'attribut de métadonnée `_use_local_application_attributes` est spécifié alors que le fichier `attributes/local_application_attributes.adoc` n'existe pas, le rendu du livre échouera.

Il ne doit pas y avoir de ligne vide tant que le fichier `run_app.adoc` n'a pas été inclus.

7. Les attributs de métadonnées

7.1. Qu'est-ce que sont les attributs de métadonnées ?

Les attributs sont des noms auxquels il est possible d'associer une valeur. En utilisant le nom, vous accédez à la valeur associée.

Les attributs de métadonnées sont des attributs qui contiennent des informations sur un livre, un chapitre ou un exercice. Ils sont définis par Asciidocpro et ne font pas partie du langage Asciidoc.

Par exemple, l'attribut `_author` qui permet de définir l'auteur est une métadonnée d'un livre, d'un chapitre ou d'un exercice.

Les attributs de métadonnées sont définis dans le fichier `main.adoc` **avant le titre**.

7.2. Liste des attributs de métadonnées

Voici la liste complète des attributs de métadonnées :

- **attributs de métadonnées utilisables dans un livre, un chapitre ou un exercice**
  - `_author` : permet de spécifier l'auteur
  - `_duration` : permet de spécifier la durée de réalisation (pour un livre, il faut faire le total de toutes les durées des parties qui le composent)
  - `_duration_of_correction` : permet de spécifier la durée de correction (s'ajoute à la durée de réalisation pour avoir une estimation du temps total)
  - `_version_number` : numéro de version
  - `_version_date` : date de la version

Ces attributs ne sont pas utilisables dans le contexte d'un livre marqué par l'attribut `_book` et la valeur `lonely`. Effectivement, ce sont les attributs de métadonnées définis au niveau du chapitre ou de l'exercice qui seront utilisés.

- **attributs de métadonnées utilisables seulement dans un livre**
  - `_book` : permet de spécifier que le fichier `main.adoc` concerne un livre. Cet attribut peut être déclaré avec la valeur `lonely` lorsque le "livre" ne contiendra qu'un seul chapitre (avec ou sans exercices) ou un seul exercice (voir la documentation). **Cet attribut est obligatoire.**
  - `_use_local_application_attributes` : permet de spécifier que des attributs d'application sont définis au niveau d'un livre.
  - `_use_local_content_attributes` : permet de spécifier que des attributs de contenu sont définis au niveau d'un livre>>.
- **attribut de métadonnées utilisable seulement dans un chapitre**
  - `_chapter` : permet de spécifier que le fichier `main.adoc` concerne un chapitre. **Cet attribut est obligatoire.**
- **attribut de métadonnées utilisable seulement dans un exercice**
  - `_exercise` : permet de spécifier que le fichier `main.adoc` concerne un exercice. **Cet attribut est obligatoire.**

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi dans le dossier `bonus` du projet Asciidocpro.

Ce fichier met à votre disposition 3 live templates qui génèrent automatiquement les attributs de métadonnées avec les autres instructions à écrire en début de fichier :

- live template `start_chapter` pour générer les instructions de début du fichier `main.adoc` d'un chapitre. Le curseur est automatiquement positionné là où écrire le titre du chapitre.

- live template `start_exercise` pour générer les instructions de début du fichier `main.adoc` d'un exercice. Le curseur est automatiquement positionné là où écrire le titre de l'exercice.
- live template `start_book` pour générer les instructions de début du fichier `main.adoc` d'un livre. Le curseur est automatiquement positionné là où écrire le titre du livre.

7.3. Où sont affichées les valeurs des attributs de métadonnées ?

Les valeurs des attributs de métadonnées sont **affichées sous le titre du livre, du titre du chapitre, du titre de l'exercice** au niveau duquel elles sont définies.

Un attribut déclaré, mais sans valeur, ne sera pas affiché dans le document.

Cela est équivalent à ne pas écrire l'attribut.

Il y a une exception pour l'attribut `_author` : si l'attribut `_author` est déclaré sans valeur, Asciidocpro utilise la valeur de l'attribut d'application `_user_default_author` si celle-ci a été définie par l'utilisateur.

Il est possible de désactiver l'affichage des métadonnées sous le titre de chaque chapitre et exercice, il faut paramétrer la valeur de l'attribut d'application `_user_show_metadata` en lui affectant la valeur `0` (voir paramétrer Asciidocpro avec les attributs d'application.

8. Les attributs de contenu

8.1. Qu'est-ce que sont les attributs de contenu ?

Les attributs sont des noms auxquels il est possible d'associer une valeur. En utilisant le nom, vous accédez à la valeur associée.

Les attributs de contenu sont des attributs utilisés **dans le contenu d'un chapitre ou d'un exercice**. Cela permet de mettre à jour facilement certaines valeurs sans avoir à les rechercher dans le contenu.

Les attributs de contenu doivent toujours être définis après que le fichier `run_app.adoc` ait été inclus, soit après le titre et avant le contenu à écrire.

Imaginez que vous réalisez un support de formation sur le traitement de texte Writer de la suite bureautique Libre Office. Vous avez besoin de mentionner plusieurs fois le numéro de la version utilisée dans votre support. Lorsque le numéro de version va changer, il vous faudra mettre à jour le numéro de version partout où il a été mentionné. C'est source d'erreur.

Il est préférable d'utiliser un attribut de contenu :

```
:_chapter:
[[chapitre_le_traitement_de_texte_writer_de_livre_office]]
= Le traitement de texte Writer de Livre Office
include::../../../run_app.adoc[]

//attributs de contenu
:re_libre_office_version: 24.2.0 ①

// ... contenu du chapitre
Pour cette formation, nous utiliserons la version {re_livre_office_version}. ②

// ... du contenu

Vérifiez que votre version est au moins la version {re_livre_office_version}. ②
```

① Définition de l'attribut de contenu avant le contenu, mais après le titre principal (une fois que le fichier `run_app.adoc`) a été inclus.

② affichage de la valeur de l'attribut de contenu

Déclarer les attributs de contenu avant le contenu du chapitre ou de l'exercice facilite leur mise à jour.

Il est tout à fait possible de définir des attributs de contenu sous le titre d'un livre avant d'ajouter les chapitres et ou les exercices. Ceux-ci seront dont utilisables dans tous les éléments ajoutés.

Il n'y a aucune limite sur le nombre d'attributs de contenu utilisable dans un support.

Si vous avez de nombreux attributs de contenu et que vous souhaitez les "sortir" du livre, du chapitre ou de l'exercice, vous pouvez créer un dossier `attributes` dans le dossier contenant le fichier `main.adoc`.

Voici un exemple depuis un chapitre :

```
chapters
 │
 ├──────suite_libre_office
 │    ├──────calc
 │    │       main.adoc
 │    │
 │    └──────writer
 │         │   main.adoc
 │         │
 │         └──────attributes
 │                    mes_attributs_de_contenu.adoc ①
```

① Fichier qui contient les attributs de contenu

Pour accéder aux attributs depuis le fichier `main.adoc`, il suffit d'inclure le fichier qui contient les

attributs :

```
:_chapter:
[[chapitre_le_traitement_de_texte_writer_de_livre_office]]
= Le traitement de texte Writer de Livre Office
include::../../../run_app.adoc[]

//inclusion du fichier des attributs de contenu
\include::attributes/mes_attributs_de_contenu.adoc[] ①

// ... contenu du chapitre
Pour cette formation, nous utiliserons la version {re_livre_office_version}.

// ... du contenu

Vérifiez que votre version est au moins la version {re_livre_office_version}. ...
```

① Inclusion du fichier qui contient les attributs de contenu du chapitre.

Si vous souhaitez partager des attributs entre différents chapitres, exercices ou livres, vous pouvez lire les parties sur le partage de composants entre des chapitres de thèmes différents, le partage de composants entre des chapitres du même thème, le partage de composants entre des exercices d'un même chapitre.

9. Adopter les conventions rédactionnelles d'Asciidocpro

9.1. Pourquoi suivre les conventions rédactionnelles d'Asciidocpro ?

L'objectif d'Asciidocpro est de "standardiser" le contenu d'un support.

Ainsi, lorsque plusieurs rédacteurs adoptent les mêmes conventions, il devient facile :

- d'intégrer à son travail des parties rédigées par d'autres personnes
- de partager son travail avec d'autres personnes qui ont l'assurance d'utiliser les mêmes conventions

L'intérêt majeur est le travail à plusieurs. Cela permet également d'adopter une convention qui a été réfléchie pour être prise en charge par Asciidocpro.

Enfin, l'adoption des conventions d'Asciidocpro vous assure de pouvoir utiliser toutes les fonctionnalités d'AsciidocproM. Effectivement, ce logiciel qui permet de manager un projet Asciidocpro exploite les conventions recommandées pour analyser le contenu des fichiers, faire les mises à jour, etc.

Voici quelques exemples de ce que peut faire AsciidocproM lorsque les conventions rédactionnelles sont respectées :

- générer une liste des compétences par chapitre
- générer une liste des mots clés par chapitre
- générer une liste des chapitres par mot clé
- générer un dictionnaire des mots clés
- générer une liste des liens entrants d'un chapitre / exercice / livre
- générer une liste des liens sortants d'un chapitre / exercice / livre
- générer une liste des compétences par chapitres / exercices / livre
- etc.

Les conventions de notation peuvent être facilement respectée en adoptant l'utilisation delive templates. D'ailleurs, un fichier de live templates est fourni en bonus avec chaque version d'Asciidocpro. Il n'y a plus qu'à les importer dans votre IDE Jetbrains (PhpStorm, IntelliJ, etc.).

9.2. Convention de notation d'une question

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.

Si vous voulez poser une question, faites-le en adoptant ce morceau de code :

```
[.question] ①
**** ②
*Q{counter:_question})* ③
④
//end question ⑤
**** ⑥
```

① rôle du bloc : permet d'identifier le bloc compris entre les `*` comme étant une question.

② début du bloc de la question

③ lettre `Q` pour "question" suivie du numéro de celle-ci. C'est le processeur asciidoc qui gère l'incrémentation de la valeur de l'attribut `_question`. Le mot `counter:` indique au processeur qu'il faut incrémenter la valeur courant de l'attribut `_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 questions.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `q` pour générer le code d'une question. Le curseur est automatiquement positionné là où écrire la question.

9.3. Convention de notation d'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
//end answer
**** ⑥
//end _show_correction ⑦
endif::[] ⑧
```

① test qui évalue s'il faut ou non afficher la réponse.

② type de bloc : 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.

Vous pouvez définir le comportement global de l'application concernant l'affichage des réponses. Par défaut, les réponses sont affichées. Pour les masquer, cela peut être fait au niveau global ou au niveau d'un livre en affectant la valeur `0` à l'attribut d'application `_user_show_correction`

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `r` pour générer le code d'une réponse. Le curseur est automatiquement positionné là où écrire la réponse.

9.4. Convention de notation des mots clés

Les mots clés sont des mots plus importants que les autres. Il est important de les identifier clairement.

9.4.1. Marquer un mot comme étant un mot clé

Lorsque vous avez un mot que vous considérez comme "clé" dans une ligne de votre fichier asciidoc, vous pouvez le "marquer" de la façon suivante :

```
[.keyword]#((mot clé))# ① ②
```

① `[.keyword]` est un rôle qui porte sur le contenu compris entre `#`. Les `#` doivent encadrer le mot clé qui peut faire plusieurs mots. Ce rôle peut faire l'objet d'une personnalisation dans le fichier de thème.

② les doubles parenthèses `((` et `))` encadrent le mot clé pour qu'il soit ajouté automatiquement à l'index. Ce serait dommage de s'en priver (d'autant plus que vous pouvez désactiver l'index en configurant le bon attribut).

Grâce à ce marquage, vos mots clés sont clairement identifiés et AsciidocproM sera en mesure de les extraire.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, au moment d'écrire un mot clé, vous pouvez utiliser le live template `kk` pour générer le code marquant le contenu comme un mot clé. Le curseur est automatiquement positionné à l'endroit où écrire l'expression clé.

Si le mot clé est déjà écrit, vous pouvez le sélectionner et choisir le live template `kk_surround`. Ce live template est mobilisable sur un contenu sélectionné. Après sélection, CTRL+ALT+J et choisir `kk_surround`. La sélection sera marquée comme un mot clé.

9.4.2. Créer une ancre sur un mot clé

Si vous souhaitez créer une ancre sur un mot clé, vous pouvez le faire en précédent la déclaration d'un mot clé vue précédemment par une ancre dont l'identifiant commence obligatoirement par `keyword_` (ne pas oublier l'underscore après le mot "keyword" !) Ensuite, vous ajoutez au préfixe `keywork_` le mot clé.

Je conseille d'écrire l'ancre avec la notation `snake_case` (exemple : `keyword_voici_un_mot_cle_d_un_cours`. C'est-à-dire que les espaces, apostrophes, etc, sont remplacés par des underscores.

Exemple avec l'expression "langage Asciidoc" qui doit être marquée comme un mot clé et identifiée :

```
[[keyword_langage_asciidoc]][.keyword]#((langage Asciidoc))# est un langage de
balisage léger proche du langage Markdown,...
```

Le fait de préfixer l'identifiant de l'ancre par `keyword_` permet de profiter de l'autocomplétion de l'IDE pour créer facilement un lien sur une ancre de mot clé.

Un mot clé ne doit être marqué qu'une seule fois avec la même ancre / identifiant. C'est-à-dire que cet identifiant doit être unique que ce soit dans les chapitres et les exercices, vous ne devez l'utiliser qu'une seule fois.

Cela s'explique par le fait qu'un identifiant est toujours unique et que s'il faut faire un lien vers celui-ci, vous ne pouvez pas faire correspondre ce lien vers plusieurs ancres !

Si jamais vous tenez absolument à donner plusieurs définitions à un même mot clé, alors vous devez explicitement rendre unique l'identifiant utilisé.

Imaginons que vous utilisiez le mot clé "mot clé" dans le chapitre A. Vous l'identifiez comme tel de la façon suivante `[[keyword_mot_clé]][.keyword]#((mot clé))#`.

Vous écrivez un chapitre B et vous donnez une nouvelle définition du même mot clé, il vous faut rendre unique l'identifiant de la manière suivante `[[keyword_mot_clé_2]][.keyword]#((mot clé))#`.

Les identifiants de l'ancre `keyword_mot_clé` et `keyword_mot_clé_2` sont uniques.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, au moment d'écrire un mot clé, vous pouvez utiliser le live template `kkk` pour générer le code marquant le contenu comme un mot clé **en lui ajoutant une ancre**. Le curseur est automatiquement positionné à l'endroit où écrire l'expression clé.

Si le mot clé est déjà écrit, vous pouvez le sélectionner et choisir le live template `kkk_surround`. Ce live template est mobilisable sur un contenu sélectionné. Après sélection, faire CTRL+ALT+J et choisir `kkk_surround`. La sélection sera marquée comme un mot clé avec une ancre.

9.4.3. Un mot clé avec sa définition

Si le mot clé est placé dans un paragraphe contenant sa définition, vous pouvez spécifier le rôle du paragraphe comme étant une définition.

Imaginez que durant la rédaction de votre support, vous décidiez d'expliquer ce qu'est Asciidoc. Vous écrivez le contenu suivant :

```
// ... du contenu tout plein !

La documentation peut être écrite avec le langage Asciidoc. ①

[.definition] ② ③
Le [[keyword_langage_asciidoc]][.keyword]#((langage Asciidoc))# est un langage de
balisage léger proche du langage Markdown, proposant une richesse sémantique similaire
à DocBook. Un document écrit en AsciiDoc forme déjà un document lisible par des
humains tout en étant interprétable par des programmes. L'utilisation d'un éditeur de
texte permet de créer et éditer un document AsciiDoc. ④
⑤
Un document AsciiDoc peut être publié dans de nombreux formats de sortie, notamment
HTML, PDF, DocBook, EPUB et Unix manpages. ⑥

// ... encore du contenu
```

① du contenu divers et varié qui précède l'explication ou la définition d'un mot clé

② le rôle du bloc `[.definition]` **placé avant le paragraphe contenant la définition** permet à Asciidocpro de savoir que le paragraphe qui suit est la définition d'un mot clé.

③ **il ne faut pas laisser de ligne vide entre le rôle et le paragraphe qui contient la définition.**

④ paragraphe de la définition du mot clé. **La définition doit contenir le mot clé** lui-même marqué par le rôle `[.keyword]#((mot clé))#`. Si le mot clé n'est par marqué comme tel dans la définition, AsciidocproM ne sera pas en mesure de lier la définition à ce mot clé. La définition sera une définition orpheline car elle ne pourra être rattachée à un mot clé. Généralement, vous proposerez une ancre sur le mot clé afin de pouvoir le pointer avec des liens. Pour cela il suffit de placer une ancre juste avant le mot clé ce qui donne `[[keyword_mot_cle]][.keyword]#((mot clé))#`

⑤ la ligne vide indique à Asciidocpro que la définition est terminée.

⑥ suite du contenu

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, avant le paragraphe de la définition, vous pouvez utiliser le live template `def` pour ajouter le nom du bloc qui indique une définition contenu dans une phrase (doit être appelé sur la ligne précédent la phrase de définition).

9.5. Convention de notation des compétences

Si vous êtes dans le cas où vous listez des compétences dans des chapitres ou des exercices, vous pouvez les identifier de façon explicite avec le nom de bloc `skill` (au singulier).

Cela peut être utile pour repérer rapidement les compétences ciblées par un chapitre ou un exercice que cela soit dans l'un de vos supports ou celui d'un collaborateur. AsciidocproM est d'ailleurs capable de récupérer les compétences identifiées avec ce rôle.

Voici la convention à adopter pour déclarer des compétences :

```
[.skill] ①
**** ②
* phrase simple exprimant une compétence ③
* une autre phrase simple exprimant une autre compétence③
//end skill ④
**** ⑤
```

① Rôle du bloc permettant d'identifier le contenu, ici une liste de compétences.

② Début du bloc de compétences

③ Compétence abordée. **Il ne doit y avoir qu'une seule compétence par phrase ou item**.

④ commentaire indiquant la fin de la liste des compétences. C'est utile (mais pas obligatoire) pour voir la fin des compétences.

⑤ Fin du bloc contenant les compétences

Il n'est pas obligatoire de lister les compétences dans des items de liste. L'exemple suivant est tout aussi valable :

```
[.skill]
****
Ceci est une phrase simple exprimant une compétence.
①
Ceci est une autre phrase simple exprimant une autre compétence
****
```

① Chaque compétence doit être séparée par une ligne vide (un commentaire n'est pas une ligne vide).

Vous pouvez déclarer autant de blocs `[.skill]` que vous voulez dans un fichier. Je conseille tout de même de les centraliser en début de fichier.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `todo` pour générer le code du bloc contenant une liste de tâches à faire par rapport au contenu.

9.6. Convention de notation d'une note 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
[.noteprof] ②
**** ③
④
//end noteprof ⑤
**** ⑥
//end _show_note_prof ⑦
endif::[] ⑧
```

① test qui évalue s'il faut ou non afficher la note

② rôle du bloc permettant d'identifier le bloc comme étant une note pour le professeur

③ délimiteur du début du bloc contenant le texte de la note

④ zone dans laquelle écrire la note pour le professeur. N'importe quel contenu peut être écrit ici.

⑤ 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.

⑥ délimiteur de fin de la note

⑦ commentaire indiquant la fin du test. C'est utile pour s'y retrouver visuellement lorsque la note contient d'autres tests.

⑧ fin du test d'affichage de la note

Vous pouvez définir le comportement global de l'application concernant l'affichage des notes du professeur. Par défaut, les notes sont affichées. Pour les masquer, cela peut être fait au niveau global ou au niveau d'un livre en affectant la valeur `0` à l'attribut d'application `_user_show_note_prof`

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `note_prof` pour générer le code d'une note destinée au professeur. Le curseur est automatiquement positionné là où écrire la note.

9.7. Convention de notation d'une liste de tâches à faire

Ce bloc de code est utile lorsque vous voulez noter ce qu'il reste à faire dans le fichier dans lequel vous rédigez le contenu. Il faut voir cela comme un mémo, c'est-à-dire une note que vous pouvez écrire pour vous souvenir de faire telle ou telle chose. C'est également utile en cas de partage d'un contenu avec un collaborateur. Vous-même et ce dernier savez ce qu'il reste à faire.

Voici le bloc de code correspondant :

```
ifeval::[{_show_todo}==1] ①
[.todo] ②
**** ③
* Ajouter une partie sur la première pompe sur un doigt de Chuck ④
* Multiplier les poids utilisés par 50 dans le contenu ④
**** ⑤
//end eval _show_content_todo ⑥
endif::[] ⑦
```

① test qui contrôle l'affichage de la liste des choses à faire

② rôle du bloc permettant d'identifier le contenu, ici une liste de tâche à réaliser.

③ délimiteurs `**` indiquant le début du bloc

④ tâche à réaliser. **Il ne doit y avoir qu'une seule tâche par ligne**. Une tâche doit être une phrase simple.

⑤ délimiteurs `**` indiquant la fin du bloc

⑥ commentaire permettant d'identifier la fin du test de l'affichage de la liste des tâches

⑦ fin du test qui contrôle l'affichage de la liste des choses à faire.

Il n'est pas obligatoire de lister les tâches à faire dans des items de liste. L'exemple suivant est tout aussi valable :

```
[.todo]
****
Ceci est une phrase simple exprimant une tâche à faire.
①
Ceci est une autre phrase simple exprimant une autre tâche à faire
****
```

① Chaque compétence doit être séparée par une ligne vide (un commentaire n'est pas une ligne vide).

Vous pouvez déclarer autant de blocs `[.todo]` que vous voulez dans un fichier. Je conseille tout de même de les centraliser en début de fichier.

Si vous utilisez un IDE Jetbrains pour écrire vos contenus Asciidoc, vous pouvez

récupérer les live templates Asciidocpro prêts à l'emploi.

Ensuite, vous pouvez utiliser le live template `todo` pour générer le code du bloc contenant une liste de tâches à faire par rapport au contenu.

10. Utiliser des live templates

10.1. C'est quoi un live template ?

Des live templates sont des morceaux de code qu'il est possible de générer en écrivant quelques caractères.

Par exemple, vous écrivez `start_chapter` et le contenu suivant est généré automatiquement :

```
//:_author:
:_duration:
:_duration_of_correction:
:_version_number:
:_version_date:
:_chapter:
[[chapitre_]]
= ①
include::../../../run_app.adoc[]
```

① Le curseur se positionne automatiquement au niveau du titre et vous n'avez plus qu'à saisir ce dernier. Le contenu de l'ancre est automatiquement généré et vous n'avez pas à vous soucier du code à écrire pour inclure le fichier `start_chapter.adoc`.

Asciidocpro est livré avec un fichier de templates pour les IDE Jetbrains. Ces lives templates permettent de :

- générer le début d'un livre avec plusieurs chapitres et ou exercices
- générer le début d'un livre avec un seul chapitre ou exercice
- générer le début d'un chapitre
- générer le début d'un exercice
- générer un titre de chapitre
- générer un titre d'exercice
- générer les attributs d'application
- générer un lien vers une ancre
- générer un template de question
- générer un template de réponse
- générer un template d'une note du professeur
- générer des marqueurs (de compétences, de mots clés, etc.)

- etc.

Le point suivant aborde leur "installation".

10.2. Des live templates dédiés à Asciidocpro

Si vous êtes utilisateur d'un IDE Jetbrains et que vous utilisez le plugin gratuit Asciidoc, je vous conseille d'adopter les live templates fournis avec le projet Asciidocpro.

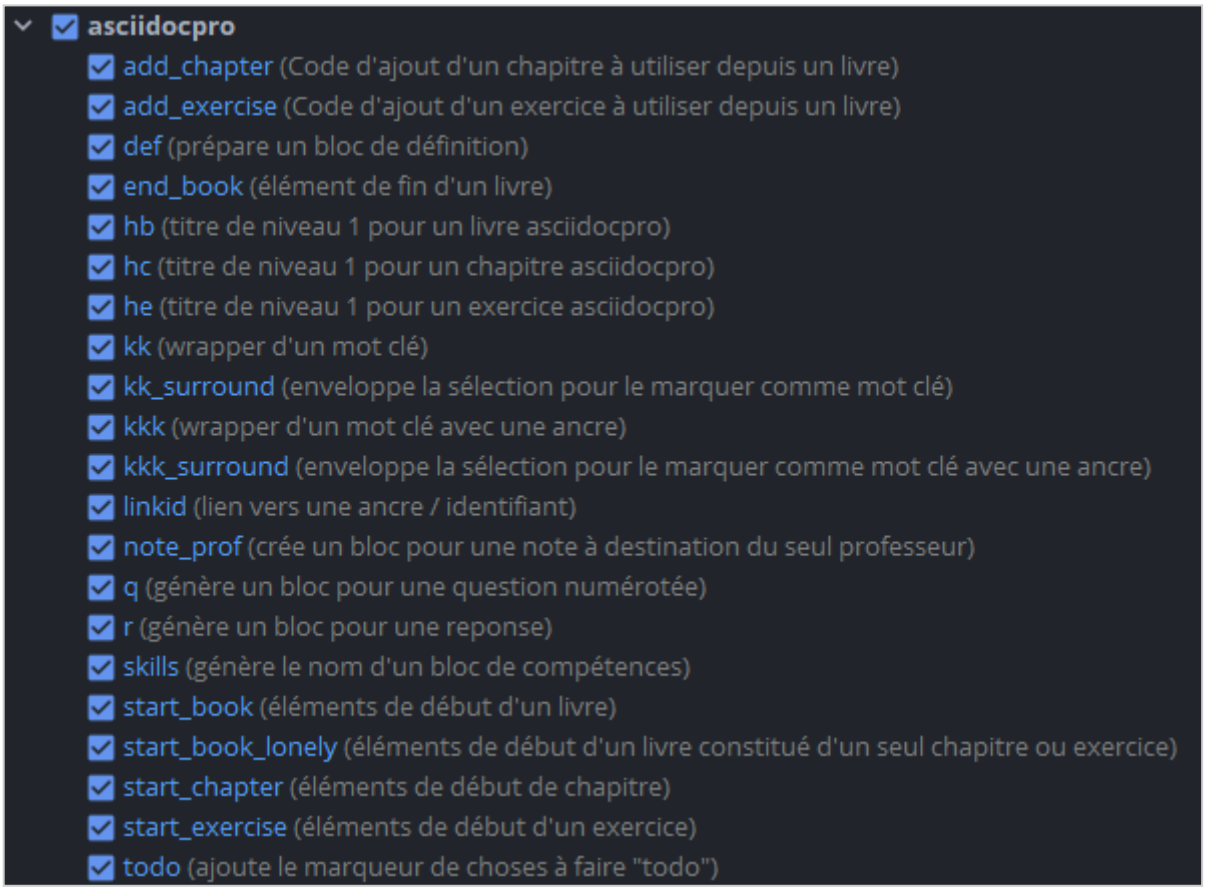
A chaque version d'Asciidocpro est associé un fichier `asciidocpro.xml` (dans le dossier `bonus`).

Pour importer les live templates de ce fichier, vous devez le placer sur votre machine dans le dossier équivalent à ce chemin :

```
C:\Users\nom_utilisateur\AppData\Roaming\JetBrains\PhpStorm2023.2\templates
```

L'exemple est donné ici avec la version 2023.2 de PhpStorm, mais le fichier peut tout à fait être utilisé depuis un autre IDE Jetbrains (par exemple IntelliJ).

Voici la liste des live templates Asciidocpro au 26/02/24 :



Si vous êtes utilisateur de Visual Studio Code, l'équivalent d'un live template est un "snippet".

11. Définir une image de couverture

11.1. Utiliser une image de couverture par défaut pour tous les pdf

Pour définir une image de couverture qui sera utilisée par défaut pour tous les pdf générés, il faut définir l'attribut d'application `_user_cover_image_filename` au niveau global, soit dans le fichier `config/attributes/global_application_attributes.adoc`

```
//fichier config/attributes/global_application_attributes.adoc
//
// ... d'autres attributs d'application
:_user_cover_image_filename: une_image.png ①
```

① L'attribut `_user_cover_image_filename` contient le nom de l'image à utiliser.

L'image doit être placée dans le dossier `config/images` du projet Asciidocpro.

Attention, dans un fichier qui contient des attributs d'application, il ne faut aucune ligne vide, y compris en fin de fichier.

11.2. Utiliser une image de couverture spécifique à un livre

Pour définir une image de couverture qui sera utilisée pour un livre spécifique, il faut définir l'attribut d'application `_user_cover_image_filename` au niveau du dossier du livre dans le fichier `attributes/local_application_attributes.adoc`

```
//fichier attributes/local_application_attributes.adoc
//
// ... d'autres attributs d'application
:_user_cover_image_filename: image_specifique_au_livre.png ①
```

① L'attribut `_user_cover_image_filename` contient le nom de l'image à utiliser.

L'image qui doit être utilisée doit être placée dans le dossier `images` du livre concerné.

Attention, dans un fichier qui contient des attributs d'application, il ne faut aucune ligne vide, y compris en fin de fichier.

Si l'image définie au niveau d'un livre a le même nom que l'image définie au niveau global, c'est l'image de niveau global qui est utilisée comme image de couverture.

12. Définir une image de fond sur toutes les pages

Version 1 | Dernière mise à jour : 06/03/24 à 00:19

12.1. Utiliser une image de fond par défaut pour tous les pdf

Pour définir une image de fond sur toutes les pages des pdf (sauf la page de couverture), il faut définir l'attribut d'application `_user_background_image_for_all_pages` au niveau global, c'est-à-dire dans le fichier `config/attributes/global_application_attributes.adoc`.

```
//fichier config/attributes/global_application_attributes.adoc
//
// ... d'autres attributs d'application
:_user_background_image_for_all_pages: une_image_de_fond.png ①
```

① L'attribut `_user_background_image_for_all_pages` contient le nom de l'image à utiliser sur toutes les pages.

L'image doit être placée dans le dossier `config/images` du projet Asciidocpro.

Attention, dans un fichier qui contient des attributs d'application, il ne faut aucune ligne vide, y compris en fin de fichier.

12.2. Utiliser une image de fond pour un livre spécifique

Pour définir une image de couverture qui sera utilisée pour un livre spécifique, il faut définir l'attribut d'application `_user_background_image_for_all_pages` au niveau du dossier du livre dans le fichier `attributes/local_application_attributes.adoc`.

```
//fichier attributes/local_application_attributes.adoc
//
// ... d'autres attributs d'application
:_user_background_image_for_all_pages: image_de_fond_specifique_au_livre.png ①
```

① L'attribut `_user_background_image_for_all_pages` contient le nom de l'image à utiliser.

L'image qui doit être utilisée sur toutes les pages doit être placée dans le dossier `images` du livre concerné.

Attention, dans un fichier qui contient des attributs d'application, il ne faut aucune ligne vide, y compris en fin de fichier.

Si l'image définie au niveau d'un livre a le même nom que l'image définie au niveau global, c'est l'image de niveau global qui est utilisée comme image de couverture.

13. Injecter automatiquement une page d'entête pour un "livre"

Asciidocpro vous permet d'injecter automatiquement une page après la table des matières d'un "livre". Cette page est créée une fois et sera utilisable dans tous les livres.

Pour illustrer cette fonctionnalité, vous allez préparer une page qui va contenir le copyright à utiliser dans tous les "livres" générés.

Voici les étapes à suivre :

1. Créez un fichier `book_template.adoc` (le nom du fichier est libre) dans le dossier `config/templates/`. Si le dossier n'existe pas, créez-le.
2. Dans le fichier `config/attributes/global_application_attributes.adoc`, ajoutez l'attribut `_user_header_book_template_filename` et affectez-lui pour valeur le nom du fichier créé :

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

① le fichier d'entête à utiliser pour un livre est `book_template.adoc`. Asciidocpro le cherchera automatiquement dans `config/templates/`

Il ne faut surtout pas laisser de ligne vide dans le fichier `config/attributes/global_application_attributes.adoc` même en fin de fichier.

3. Préparez le contenu du fichier `config/templates/book_template.adoc` (à adapter en fonction du nom que vous avez choisi) :

```
[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).

4. Une fois votre fichier créé, il faut indiquer qu'il doit être affiché en configurant l'attribut d'application `_user_show_header_templates` dans le fichier `config/attributes/global_application_attributes.adoc` :

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

① tous les fichiers d'entête définis seront affichés. Pour rappel, il est possible de définir les attributs d'application au niveau d'un livre, l'exemple utilise une configuration au niveau global

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

Grâce à Asciidocpro, votre fichier d'entête sera également rendu dans la prévisualisation.

14. Injecter automatiquement une page d'entête à chaque chapitre

Asciidocpro vous permet d'injecter automatiquement une page après le titre d'un chapitre d'un livre. Cette page est créée une fois et sera injectée dans tous les chapitres d'un livre.

Le principe d'injection est le même que pour un "livre".

Pour illustrer cette fonctionnalité, vous allez préparer une page qui va contenir une note à lire avant chaque chapitre.

Voici les étapes à suivre :

1. Créez un fichier `chapter_template.adoc` (le nom du fichier est libre) dans le dossier `config/templates/`. Si le dossier n'existe pas, créez-le.
2. Dans le fichier `config/attributes/global_application_attributes.adoc`, ajoutez l'attribut `_user_header_chapter_template_filename` et affectez-lui pour valeur le nom du fichier créé :

```
//autres attributs
// ...
:_user_header_book_template_filename: book_template.adoc ①
:_user_header_chapter_template_filename: chapter_template.adoc ②
```

① un fichier d'entête de livre est déjà spécifié dans notre exemple.

② le fichier d'entête à utiliser pour un livre est `chapter_template.adoc`. Asciidocpro le cherchera automatiquement dans `config/templates/`

Il ne faut surtout pas laisser de ligne vide dans le fichier `config/attributes/global_application_attributes.adoc` même en fin de fichier.

3. Préparez le contenu du fichier `config/templates/chapter_template.adoc` (à adapter en fonction du nom que vous avez choisi) :

```
[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.
====
```

4. Une fois votre fichier créé, il faut indiquer qu'il doit être affiché en configurant l'attribut d'application `_user_show_header_templates` dans le fichier `config/attributes/global_application_attributes.adoc` :

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

① tous les fichiers d'entête définis seront affichés (donc ici, celui au niveau du livre et celui pour chaque chapitre). Pour rappel, il est possible de définir les attributs d'application au niveau d'un livre, l'exemple utilise une configuration au niveau global

A partir de maintenant, tous vos chapitres contiendront votre fichier d'entête sans que vous ayez à faire quoi que ce soit !

Grâce à Asciidocpro, votre fichier d'entête sera également rendu dans la prévisualisation.

15. Injecter automatiquement une page d'entête à chaque exercice

Asciidocpro vous permet d'injecter automatiquement une page après le titre d'un exercice d'un livre. Cette page est créée une fois et sera injectée dans tous les exercices d'un livre.

Le principe d'injection est le même que pour un chapitre.

Pour illustrer cette fonctionnalité, vous allez préparer une page qui va contenir une note à lire avant chaque exercice.

Voici les étapes à suivre :

1. Créez un fichier `exercise_template.adoc` (le nom du fichier est libre) dans le dossier `config/templates/`. Si le dossier n'existe pas, créez-le.
2. Dans le fichier `config/attributes/global_application_attributes.adoc`, ajoutez l'attribut `_user_header_chapter_template_filename` et affectez-lui pour valeur le nom du fichier créé :

```
//autres attributs
// ...
:_user_header_book_template_filename: book_template.adoc ①
:_user_header_chapter_template_filename: chapter_template.adoc ①
:_user_header_exercise_template_filename: exercise_template.adoc ②
```

① des fichiers d'entête de livre et de chapitre sont déjà spécifiés dans notre exemple.

② le fichier d'entête à utiliser pour un exercice est `exercise_template.adoc`. Asciidocpro le cherchera automatiquement dans `config/templates/`

Il ne faut surtout pas laisser de ligne vide dans le fichier `config/attributes/global_application_attributes.adoc` même en fin de fichier.

3. Préparez le contenu du fichier `config/templates/exercise_template.adoc` (à adapter en fonction du nom que vous avez choisi) :

```
[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.

Il est important de respecter la durée de réalisation indiquée sous le titre de
l'exercice.

Bon courage.
====
```

4. Une fois votre fichier créé, il faut indiquer qu'il doit être affiché en configurant l'attribut d'application `_user_show_header_templates` dans le fichier `config/attributes/global_application_attributes.adoc` :

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

① tous les fichiers d'entête définis seront affichés (donc ici, celui au niveau du livre, de chaque chapitre et exercice). Pour rappel, il est possible de définir les attributs d'application au niveau d'un livre, l'exemple utilise une configuration au niveau global

A partir de maintenant, tous vos exercices contiendront votre fichier d'entête sans que vous ayez à faire quoi que ce soit !

Grâce à Asciidocpro, votre fichier d'entête sera également rendu dans la prévisualisation.

16. 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 son contenu (notamment les chemins vers les éléments inclus).

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
│     │     │     │
│     │     │     ├──────images
│     │     │     │       image_125.png
│     │     │     │
│     │     │     └──────templates
│     │     │           les_5_regles_d_or.adoc
│     │     │
│     │     └──────chapitre_2
│     │         └──────main.adoc
│     ├──────theme_B
│     │     ├──────chapitre_alpha
│     │     │     │   main.adoc
│     │     │     │
│     │     │     └──────images
│     │     │           image_125.png
│     │     │
│     │     └──────chapitre_beta
│     │         │   main.adoc
│     │         │
│     │         └──────templates
│     │               les_5_regles_d_or.adoc
│     │
│     └──────theme_C
└──────config
```

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 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, 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 :

```
├──────books
├──────chapters
│     ├──────theme_A
│     │     ├──────chapitre_1 ②
│     │     │       main.adoc
│     │     │
│     │     └──────chapitre_2
│     │             └──────main
│     ├──────theme_B
│     │     ├──────chapitre_alpha ②
│     │     │       main.adoc
│     │     │
│     │     └──────chapitre_beta ②
│     │             main.adoc
│     │
│     ├──────theme_C
│     └──────_shared_components ①
│           ├──────images
│           │       image_125.png
│           │
│           └──────templates
│                   les_5_regles_d_or.adoc
│
└──────config
```

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.

17. 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 son contenu (notamment les chemins vers les éléments inclus).

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

Illustrons ce cas avec l'arborescence suivante :

```
├──────chapters
│    └──────theme_A
│        ├──────chapitre_1
│        │    │   main.adoc
│        │    │
│        │    ├──────images
│        │    │       image_125.png
│        │    │
│        │    └──────templates
│        │           les_5_regles_d_or.adoc
│        │
│        └──────chapitre_2
```

```
|            |    main.adoc
|            |
|            ├─────images
|            |        image_125.png
|            |
|            └─────templates
|                   les_5_regles_d_or.adoc
```

Les chapitres `chapitre_1` et `chapitre_2` utilisent la même image `image_125.png` et le même fichier asciidoc `les_5_regles_d_or.adoc`.

Si la mise à jour de l'image est forcément répercutée dans les deux chapitres, il vaut mieux la partager. Cela permet de n'avoir qu'une seule mise à jour à faire. Le raisonnement est identique pour le fichier `les_5_regles_d_or.adoc`.

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 concernés :

```
├─────chapters
|    └─────theme_A
|        ├─────chapitre_1  ②
|        |        main.adoc
|        |
|        ├─────chapitre_2  ②
|        |        main.adoc
|        |
|        └─────_shared_components ①
|            ├─────images
|            |        image_125.png
|            |
|            └─────templates
|                   les_5_regles_d_or.adoc
```

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 `images` et `templates`.

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 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.

18. 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 `exercises`.

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 des exercices du même chapitre :

```
├──────chapters
│    └──────theme_A
│        ├──────chapitre_1
│        │    │   main.adoc
│        │    │
│        │    └──────exercices ①
│        │        ├──────ex_A
│        │        │       main.adoc  ③
│        │        │
│        │        ├──────ex_B
│        │        │       main.adoc  ③
│        │        │
│        │        └──────_shared_components ②
│        │            ├──────images
│        │            │       image_125.png
│        │            │
│        │            └──────templates
│        │                    les_5_regles_d_or.adoc
```

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[]
```

19. Partager des composants entre des livres

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

Imaginons que vous ayez cette arborescence :

```
└──────books
    ├──────theme_X
    │    ├──────livre_sur_space_X  ①
    │    │    │   main.adoc
    │    │    │
    │    │    └──────config
    │    │        └──────attributes
    │    │                local_application_attributes.adoc ③
    │    │
    │    └──────livre_sur_twitter_qui_devient_X
    │           main.adoc
    │
    ├──────theme_Y
    │    └──────livre_sur_Y  ②
    │        │   main.adoc
    │        │
    │        └──────config
    │            └──────attributes
    │                   local_application_attributes.adoc ③
```

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 telles qu'elles sont), il peut être intéressant de factoriser les attributs dans un seul et même fichier.

Après factorisation, cela donnerait l'arborescence suivante :

```
└──────books
    ├──────theme_X
    │    ├──────livre_sur_space_X
    │    │    │   main.adoc
    │    │    │
    │    │    └──────config
    │    │         └──────attributes
    │    │                local_application_attributes.adoc ④
    │    │
    │    └──────livre_sur_twitter_qui_devient_X
    │           main.adoc
    │
    ├──────theme_Y
    │    └──────livre_sur_Y
    │        │   main.adoc
    │        │
    │        └──────config
    │            └──────attributes
    │                   local_application_attributes.adoc ④
    │
    └──────_shared_components ①
        └──────config
               └──────attributes ②
                      common_application_attributes.adoc③
```

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_atributes.adoc` contient le code mutualisé :

```
:_user_show_note_prof: 0
:_user_show_correction: 0
```

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 :

```
└──────books
    ├──────theme_X
    │    ├──────livre_sur_space_X
    │    │    │   main.adoc
    │    │    │
    │    │    └──────config
    │    │        └──────attributes
    │    │                local_application_attributes.adoc ④
    │    │
    │    └──────livre_sur_twitter_qui_devient_X
    │            main.adoc
    │
```

```
    ├──────theme_Y
    │    └──────livre_sur_Y
    │        │   main.adoc
    │        │
    │        └──────config
    │            └──────attributes
    │                   local_application_attributes.adoc ④
    │
    └──────_shared_components ①
        └──────application_profiles ②
              prof.adoc ③
              student.adoc ③
              student_with_correction.adoc ③
```

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 context 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.

Les fichiers qui contiennent des attributs d'application ne doivent jamais contenir de lignes vides sans quoi le document final ne sera pas correctement généré

20. 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 par Asciidocpro 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/attributes/global_application_attributes.adoc` (ou au niveau d'un livre) :

```
//d'autres attributs
//...
//désactivation du cache des diagrammes
:_user_enable_cache_diagrams: 0 ①
```

① La désactivation du cache devrait être temporaire, car à chaque fois que le document final est généré, toutes les images le sont également. Si le nombre d'images est important, le temps de génération peut être très long.

Attention à ne pas laisser de lignes vides dans le fichier `config/attributes/global_application_attributes.adoc`, y compris à la fin du fichier. Cela entraînerait un rendu incomplet

21. Mise à jour d'Asciidocpro

21.1. Comment suivre les mises à jour ?

Les mises à jour du framework Asciidocpro sont listées dans le fichier `changelog.adoc` situé à la racine d'un projet Asciidocpro.

Ce fichier liste pour chaque version :

- les fonctionnalités ajoutées
- les fonctionnalités modifiées
- les fonctionnalités dépréciées
- les fonctionnalités retirées
- les corrections de bogues
- les évolutions au cœur du framework
- les instructions de mise à jour du projet pour passer de la version n-1 à la version n

21.2. Mettre à jour sa version d'Asciidocpro

Les mises à jour doivent se faire en respectant ces étapes :

1. repérez votre numéro de version d'Asciidocpro. Ce numéro de version est placé dans le fichier `src/version_info.adoc`.
2. dans le fichier `changelog.adoc` situé à la racine du projet, repérez dans l'historique des versions votre numéro de version. Partez sur l'exemple du fichier `changelog.adoc` ci-dessous, si votre version actuelle est la version 2.0.0, et que vous voulez migrer vers la dernière version, vous devez trouver la ligne qui commence par votre numéro de version actuelle :

```
// fichier changelog.adoc
//
*v4.0.0 -> v4.1.0 [22/02/24 à 09:58]* ①
//... des informations sur les changements
//
*v3.0.0 -> v4.0.0 [22/02/24 à 09:58]* ①
//... des informations sur les changements
//...
//
*v2.0.0 -> v3.0.0 [22/02/24 à 10:01]* ②
//... des informations sur les changements
//...
//
```

① La ligne ne commence pas par votre numéro de version actuelle, il faut descendre vers des versions antérieures.

② La ligne commence par votre numéro de version, c'est à partir de ce bloc que vous devez appliquer les instructions de mise à jour

3. une fois que vous avez déterminé où vous êtes placé dans l'historique des versions, vous devez appliquer les instructions de mise à jour de façon à passer de votre version à la **version immédiatement supérieure**. Cela signifie que vous ne pouvez pas migrer de la version 2.0.0 vers la version 4.1.0 en une étape. Vous devez procéder version par version, soit la version 2.0.0 vers la 3.0.0 puis de la 3.0.0 vers la 4.0.0 et enfin de la 4.0.0 à la 4.1.0.

Depuis la version 4.0.0, la mise à jour d'un projet Asciidocpro a été grandement simplifiée.

Cette version a été conçue de façon à ce que l'utilisateur n'ait pas à modifier les entêtes des fichiers de livre, de chapitre et d'exercice.

Le processus de mise à jour devrait être identique ou presque entre les versions.

21.3. Quelques astuces pour des mises à jour rapides

21.3.1. Rechercher et remplacer une ligne dans plusieurs fichiers

Les utilisateurs des IDE Jetbrains sont encore favorisés (quoi qu'il doit probablement être possible de faire la même chose avec Visual Studio Code).

Voici comment rechercher et remplacer une ligne dans plusieurs fichiers.

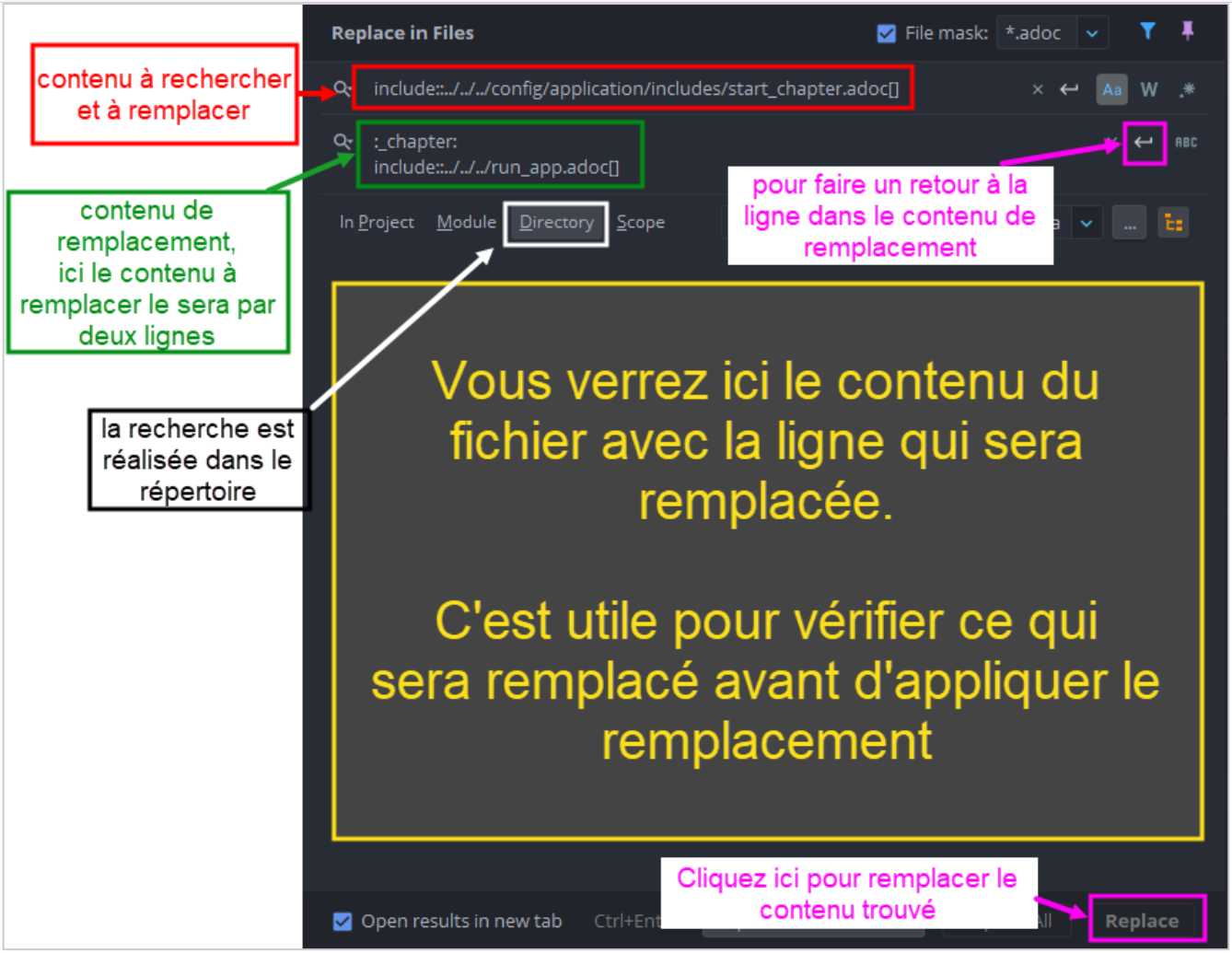
C'est très utile notamment dans le cas d'une mise à jour de ce type extraite du fichier `changelog.adoc` :

```
** Pour tous les chapitres :
*** il faut remplacer la ligne située sous le titre
`include::../../../config/application/includes/start_chapter.adoc[]` par la ligne
`include::../../../{_filename_run}[]`
*** un attribut de métadonnée `:_chapter:` doit être précisé avant le titre (ou du
moins avant la ligne `include::../../../{_filename_run}[]`)
```

Le remplacement à effectuer doit l'être dans tous les fichiers de chapitre. Faire cela a la main est fastidieux et source d'erreur.

Depuis un IDE Jetbrains, faire un clic droit sur le dossier `chapters` et choisir `Replace in Files…`.

Dans la boite de dialogue qui s'ouvre, renseignez le contenu à rechercher et ce par quoi il doit être remplacé :



Avec cette astuce, la mise à jour est très facile à appliquer !

Index

@

`_use_local_application_attributes`, 32

A

AsciiDoc, 1
atomique, 6
attribut de métadonnée, 32
attributs, 24, 32, 34
attributs de contenu, 34
attributs de métadonnées, 32
attributs d'application, 24, 31

C

chapitre, 6
compétences, 42

D

définition orpheline, 41

E

exercice, 9, 11

I

index, 24

L

live templates, 45
livre, 12

M

mots clés, 39

S

snippet, 46

V

Visual Studio Code, 46