Modèle de thèses ou de livres

Rédaction de mémoires ou d'ouvrages en Typst

12-12-2024 Template 0.5.0 Typst 0.12

Mathieu AUCEJO

Ce package Typst est une proposition de modèle pour la rédaction de mémoire thèse ou de HDR ou d'ouvrages scientifiques.

Table des matières

1. Qu'est-ce que Typst ? 1
2. Installation du modèle 2
3. Utilisation 3
  3.1. Initilisation du modèle 3
  3.2. Contenu du document 6
    3.2.1. Environnements 7
    3.2.2. Parties 7
    3.2.3. Chapitre 8
    3.2.4. Bibliographie 8
    3.2.5. Fonctions complémentaires 9
4. Paquets recommandés 10
5. Feuille de route 11
Bibliographie 12

1. Qu'est-ce que Typst ?

Typst est un nouveau langage de balise open source é crit en Rust et développé à partir de 2019 par deux étudiants allemands, Laurenz Mädje et Martin Haug, dans le cadre de leur projet de master [1,2]. La version 0.1.0 a été publiée sur GitHub le 04 avril 2023[1].

Typst se présente comme un successeur de LaTeX plus moderne, rapide et simple d'utilisation. Parmi ses avantages, on peut citer :

- la compilation incrémentale ;
- des messages d'erreur clair et compréhensible ;
- un langage de programmation Turing-complet ;
- une système de style cohérent permettant de configurer aisément tous les aspects de son document (police, pagination, marges, ...) ;
- une communauté active et sympathique (serveur Discord pour le support, annonce de nouveaux paquets) ;
- un système de paquets simple d'utilisation (pour rechercher ou voir la liste des paquets, n'hésitez pas à visiter Typst: Universe) ;

[1] Adresse du dépôt GitHub : https://github.com/typst/typst

- des extensions pour VS Code existent, comme `Typst LSP` ou `Tinymist`, pour avoir des fonctionnalités similaires à `LaTeX Workshop`.

Concernant la courbe d'apprentissage, la documentation de Typst est suffisamment bien écrite et détaillée pour permettre de créer rapidement ses propres documents. Il faut compter moins d'une heure pour prendre en main la syntaxe (sans mentir 😁). Pour accéder à la documentation, suivez ce lien. Pour faciliter la transition de LaTeX vers Typst, un guide est disponible ici.

Typst peut être utilisé comme LaTeX de deux manières différentes :

1. En ligne, via l'application web Typst. Il suffit de créer un compte pour commencer à rédiger ses documents. L'application web fonctionne de façon similaire à Overleaf. L'offre gratuite est généreuse et permet de :
   - Créer et éditer des projets ;
   - Partager et collaborer sur des projets ;
   - Convertir des fichiers LaTeX ou Word ;
   - 200 Mb de stockage et jusqu'à 100 fichiers par projet.
2. En local, en installant le compilateur Typst. Pour cela, il faut suivre les instructions contenues dans le fichier README.md du dépôt Github officiel. Une fois le compilateur installé, il est possible de compiler ses fichiers en utilisant la commande `typst compile` (compilation unitaire) ou `typst watch` (compilation incrémentale) dans le terminal.

   Une solution complémentaire consiste à utiliser un éditeur de texte comme VS Code avec l'extension `Tinymist` pour bénéficier de la coloration syntaxique, de l'autocomplétion et de la prévisualisation (munie d'une fonctionnalité similaire à `synctex` en LaTeX) et export au format PDF.

2. Installation du modèle

La collection de gabarits Cnam est disponible sur le dépôt Github de l'auteur. Vous pouvez ainsi soit cloner le dépôt, soit télécharger le fichier zip de la dernière *Release* contenant les gabarits. Pour utiliser les gabarits, deux possibilités s'offrent à vous :

1. Copier l'ensemble du dossier `cnam_templates` dans le dossier de votre projet Typst. Vous pouvez alors inclure les gabarits dans votre document en utilisant par exemple la commande :

   ```
   #import "./src/book.typ": *
   ```

   > L'adresse du fichier à utiliser dans votre fichier principal `nom_de_mon_document.typ` dépend de l'emplacement du dossier `book_template` dans votre projet.
   >
   > Il faut toutefois noter que le dossier contenant les gabarits doit être situé dans le même répertoire que votre fichier principal.

2. Installer localement le dossier `book_template` dans un dossier accessible par le compilateur Typst. Pour cela, il suffit de suivre les instructions de la documentation officielle ici. Pour cela, il faut cependant que le nom du dossier corresponde au numéro de version du paquet et que celui-ci soit contenu dans le dossier `book_template` (actuellement `book/0.5.0`).

```
// Si installation dans le dossier `/typst/packages/local`
#import "@local/book:0.5.0": *

// Si installation dans le dossier /typst/packages/preview`
#import "@preview/book:0.5.0": *
```

Ce process peut être automatisé en utilisant le script `just` contenu dans le fichier `justfile`. Pour cela, il suffit de lancer dans un terminal ouvert dans le dossier `cnam-templates` contenant les gabarits. . Une fois `just` installé, il suffit de lancer la commande :

```
# Pour installer les gabarits dans le dossier `/typst/packages/local`
just install
# Pour installer les gabarits dans le dossier `/typst/packages/preview`
just install-preview
```

> Pour installer `just`, il faut suivre les instructions figurant dans le fichier README.md du dépôt Github officiel.

3. Utilisation

Pour utiliser le modèle, il faut l'importer dans votre fichier principal `typ` en utilisant la commande suivante.

```
#import "@preview/book:0.5.0": *
```

> Si vous décomposez votre document en différents fichiers, il faut insérer la commande précédente en préambule de chaque fichier.

3.1. Initilisation du modèle

Après avoir importé le modèle, celui doit être initialisé en appliquant une règle d'affichage (`show rule`) avec la commande `#book( )` en passant les options nécessaires avec l'instruction `with` dans votre fichier principal `typ` :

```
#show book.with(
  ...
)
```

Le modèle `#book( )` possède un certain nombre de paramètres permettant de personnaliser le document. Voici la liste des paramètres disponibles :

```
#book(
  (title): "Titre du document",
```

```
  ⟨author⟩: "Nom de l'auteur",
  ⟨type⟩: "thesis",
  ⟨lang⟩: "fr",
  ⟨logo⟩: "image(\"resources/images/logo_cnam.png\")",
  ⟨body-font⟩: "Lato",
  ⟨math-font⟩: "Lete Sans Math",
  ⟨config-titre⟩: (:),
  ⟨config-colors⟩: (:)
)[⟨body⟩]
```

Argument

⟨title⟩: "Titre du document" `str`

Titre du mémoire de thèse ou de HDR.

Argument

⟨author⟩: "Nom de l'auteur" `str`

Nom de l'auteur du mémoire.

Argument

⟨type⟩: "thesis" `str`

Type de document.

Deux types de modèles sont actuellement disponibles :

- `"thesis"` pour une thèse (doctorat ou HDR)
- `"textbook"` pour un ouvrage scientifique

Argument

⟨lang⟩: "fr" `str`

Langue du document. En fonction de la valeur prise par ce paramètre, la localisation du document sera adaptée.

Outre le français, la seule langue prise en compte est l'anglais (`lang: "en"`).

Argument

⟨logo⟩: "image(resources/images/logo_cnam.png)" `content`

Chemin vers le logo de l'établissement de préparation du mémoire.

> Il faut que le template soit à la racine du répertoire pour que le chemin soit correctement interprété. Dans le cas contraire, une erreur de compilation sera générée.

Argument

⟨body-font⟩: "Lato" `str`

Nom de la police de caractère du corps du texte.

Argument

⟨math-font⟩: "Lete Sans Math" `str`

Nom de la police de caractère des équations mathématiques.

Les polices de caractère doivent être préalablement installées sur votre système pour être utilisées dans le document.

Pour vérifier la disponibilité de la police choisie, vous pouvez entrer la commande `typst font` dans un terminal.

Argument

`⟨config-titre⟩: (:)` `dictionary`

Dictionnaire permettant de personnaliser la page de titre du document.

Les options disponibles diffèrent en fonction du `type` de document choisi :

- Pour un document de `type: "thesis"` :
  - `type` : Type de document – `"phd"` ou `"hdr"`
  - `school` : Nom de l'établissement de préparation de la thèse
  - `doctoral-school` : Nom de l'école doctorale de rattachement
  - `supervisor` : Nom du/des directeurs de thèse ou du garant de la HDR
  - `cosupervisor` : Nom du ou des co-encadrants de thèse
  - `laboratory` : Nom du laboratoire de recherche de rattachement
  - `defense-date` : Date de soutenance de la thèse
  - `discipline` : Discipline dans laquelle s'inscrit la thèse
  - `speciality` : Spécialité dans laquelle s'inscrit la thèse
  - `commity` : Liste des membres du jury de soutenance

Chaque membre du jury est défini par un dictionnaire, de type `dictionary`, contenant les champs suivant :
- `name` : Nom du membre du jury
- `position` : Poste occupé (MCF, PU, ...)
- `affiliation` : Établissement de rattachement
- `role` : Rôle dans le jury (Rapporteur, Examinateur, ...)

Exemple

```
commity: (
  (
    name: "Hari Seldon",
    position: "Professeur des Universités",
    affiliation: "Streeling university",
    role: "Rapporteur",
  ),
  (
    name: "Ford Prefect",
    position: "Maître de conférences",
```

```
    affiliation: "Beltegeuse University",
    role: "Examinateur"
  ),
)
```

- Pour un document de `type: "textbook"` :
  - `subtitle` : Sous-titre de l'ouvrage
  - `edition` : Numéro de l'édition
  - `school` : Nom de l'établissement de préparation de l'ouvrage
  - `collection` : Nom de la collection
  - `year` : Année de publication
  - `cover-image` : Image de couverture de l'ouvrage

Argument

⟨config-colors⟩: (:) `dictionary`

Dictionnaire permettant de personnaliser les couleurs du document.

Les options disponibles sont les suivantes :

- `primary` : Couleur principale
- `secondary` : Couleur secondaire
- `boxeq` : Couleur des encadrés d'équations
- `header` : Couleur de l'en-tête du document

3.2. Contenu du document

Le contenu du document est à rédiger dans le fichier principal `typ` ou dans des fichiers annexes. Le modèle fournit une structure de base pour la rédaction d'un document.

D'une manière générale, la partie du fichier principal correspondant au contenu du document est structurée de la manière suivante :

```
#show: front-matter

#include "front-content.typ"

#show: main-matter

#tableofcontents()

#listoffigures()

#listoftables()
```

```
#part("Corps du document")

#include "chapitre.typ"

#bibliography("bibliography.bib")

#show: appendix

#part("Annexes du document")

#include "appendix.typ"

#back-cover(...)
```

Le contenu du mémoire est divisé en trois parties principales : `front-matter`, `main-matter` et `appendix`. Ces éléments s'accompagnent de fonctions complémentaires permettant de faciliter la rédaction du mémoire.

3.2.1. Environnements

Le modèle propose trois environnements pour structurer le contenu du mémoire :

1. **front-matter** : environnement pour le contenu préliminaire du mémoire (page de garde, résumé, remerciements, ...). En terme de pagination, les pages sont numérotées en chiffres romains et les chapitres ne sont pas numérotés. Pour activer cet environnement, il faut insérer dans le fichier principal `typ` à l'endroit souhaité la commande suivante :

   ```
   #show: front-matter
   ```

2. **main-matter** : environnement pour le contenu principal du mémoire (introduction, tables des matières chapitres, conclusion, bibliographie, ...). Les pages sont numérotées et les chapitres sont numérotés en chiffres arabes. Pour activer cet environnement, il faut insérer dans le fichier principal `typ` à l'endroit souhaité la commande suivante :

   ```
   #show: main-matter
   ```

3. **appendix** : environnement pour le contenu des annexes du mémoire. Les pages sont numérotées en chiffres romains et les chapitres sont numérotés en lettres. Pour activer cet environnement, il faut insérer dans le fichier principal `typ` à l'endroit souhaité la commande suivante :

   ```
   #show: appendix
   ```

3.2.2. Parties

Pour structurer le contenu du mémoire, il est possible de définir des parties à l'aide de la fonction `#part()`. Pour insérer une nouvelle partie, il faut insérer la commande suivante :

`#part("Titre de la partie")`

3.2.3. Chapitre

Les chapitres du mémoire sont définis par la fonction `#chapter()` qui dispose d'une certain nombre de paramètres permettant d'adapter le rendu du chapitre en fonction du contexte. Voici la liste des paramètres disponibles :

`#chapitre(⟨title⟩, ⟨abstract⟩: none, ⟨toc⟩: true, ⟨numbered⟩: true) [⟨body⟩]`

Argument
`⟨title⟩` `str`

Titre du chapitre.

Argument
`⟨abstract⟩: none` `content`

Résumé affiché sous le titre du chapitre.

Argument
`⟨toc⟩: true` `bool`

Indique si un mini table des matières doit être affichée au début du chapitre.

Argument
`⟨numbered⟩: true` `bool`

Indique si le chapitre doit être numéroté.

Exemple

```
#chapitre(
  "Introduction",
  abstract: [Résumé du chapitre],
  toc: true,
  numbered: true
)[...]
```

3.2.4. Bibliographie

Pour insérer une bibliographie, il faut insérer dans le fichier principal `typ` la commande suivante :

```
#bibliography("your-biblio-file.yml/bib")
```

> Le modèle propose deux formats de bibliographie : YAML et BibTeX.
>
> Pour le fichier YAML, celui-ci est interprété par le module `hayagriva`, dont la documentation est disponible ici.

Pour citer une référence bibliographique dans le texte, il suffit d'utiliser la commande `#cite(⟨key⟩)` ou plus simplement `@key` (où `key` est la clé associée à la référence).

Pour plus d'informations sur la gestion des références bibliographiques, il faut se référer à la documentation de la fonction `#bibliography( )` de Typst (accessible ici).

3.2.5. Fonctions complémentaires

Cette section présente les fonctions complémentaires implémentées dans le modèle pour faciliter la rédaction du mémoire.

Tables des matières

- `#tableofcontents( )` : création la table des matières.
- `#listoffigures( )` : création la table des figures.
- `#listoftables( )` : création la table des tableaux.

Figures

D'une manière générale, les figures sont insérées dans le document à l'aide de la fonction `#figure( )` de Typst. Cependant, Typst ne dispose pas actuellement de mécanismes permettant de gérer les sous-figures (numérotation et référencement). Pour pallier ce manque, le modèle intègre une fonction `#subfigure( )` permettant de gérer les sous-figures de manière adaptée. Cette fonction encapsule la fonction `#subpar.grid( )` du package `subpar`.

Exemple

```
#subfigure(
  figure(image("image1.png"), caption: []),
  figure(image("image2.png"), caption: []), <b>,
  columns: (1fr, 1fr),
  caption: [Titre de la figure],
  label: <fig:subfig>,
)
```

L'exemple précédent illustre le cas d'une figure composée de deux sous-figures. La première sous-figure est accompagnée d'une légende, tandis que la seconde possède un `label` mais pas de titre. La seconde sous-figure peut ainsi être référencée dans le texte à l'aide de la commande `@b`.

Équations

Pour encadrer une équation importante, la fonction `#boxeq( )` doit être utilisée.

Exemple

```
$
#boxeq[$p(A|B) prop p(B|A) space p(A)$]
$
```

$$p(A|B) \propto p(B|A)\, p(A) \tag{1}$$

Pour créer une équation sans numérotation, il faut utiliser la fonction #nonumeq( ).

Exemple

```
#nonumeq[$integral_0^1 f(x) dif x = F(1) - F(0)$]
```

$$\int_0^1 f(x)\,\mathrm{d}x = F(1) - F(0)$$

Quatrième de couverture

La quatrième de couverture de la thèse est générée automatiquement à partir de la fonction #back-cover( ), qui affiche les informations relatives à la thèse (titre et auteur), ainsi qu'un résumé en français et en anglais.

```
#back-cover(⟨resume⟩: none, ⟨abstract⟩: none)
```

Argument

⟨resume⟩ — content

Résumé de la thèse en français.

Argument

⟨abstract⟩ — content

Résumé de la thèse en anglais.

4. Paquets recommandés

Cette section présente une liste de paquets qui peuvent être pertinents lors de la rédaction d'un mémoire en Typst.

Dessin – CeTZ
- **Description** : Ce paquet se veut être un équivalent Typst de TiKZ pour LaTeX.
- **Liens** - Typst: Universe, dépôt GitHub et documentation.

Boîtes – showybox
- **Description** : Ce paquet permet de créer des boîtes de contenus (textes, ...) personnalisables.
- **Liens** - Typst: Universe, dépôt GitHub et documentation.

Code – codelst
- **Description** : Ce paquet permet de formatter des blocs de code source en incluant notamment la numérotation des lignes.
- **Liens** Typst: Universe, dépôt GitHub et documentation.

Algorithme – lovelace
- **Description** : Ce paquet permet d'écrire du pseudo-code, dont le formattage est personnalisable.
- **Liens** - Typst: Universe, dépôt GitHub et documentation (voir ReadMe sur GitHub).

4. Paquets recommandés

Mathématiques - `physica`

- **Description** : Ce paquet propose des raccourcis pour l'écriture de symboles mathématiques.
- **Liens** - Typst: Universe, dépôt GitHub et documentation.

Glossaire - `glossarium`

- **Description** : Ce paquet permet de créer un glossaire.
- **Liens** - Typst: Universe, dépôt GitHub et documentation.

Index - `in-dexter`

- **Description** : Ce paquet permet de créer aisément un index.
- **Liens** : Typst: Universe, dépôt GitHub et documentation.

5. Feuille de route

Le modèle est en cours de développement. Voici la liste des fonctionnalités qui sont implémentées ou le seront dans une prochaine version.

Couvertures

- [x] Page de garde
- [x] Quatrième de couverture

Environnements

- [x] Création de l'environnement `front-matter`
- [x] Création de l'environnement `main-matter`
- [x] Création de l'environnement `appendix`
- [x] Création de l'environnement `part`

Tables des matières

- [x] Création de la table des matières – `#tableofcontents()`
- [x] Création de la table des figures – `#listoffigures()`
- [x] Création de la table des tableaux – `#listoftables()`
- [x] Création d'une mini table des matières en début de chapitre grâce au paquet `minitoc` (voir lien)
- [x] Personnalisation des entrées (apparence, lien hypertexte) en modifiant l'élément `outline.entry`
- [x] Localisation des différentes tables

Figures et tableaux

- [x] Personnalisation de l'apparence des légendes des figures et des tableaux en fonction du contexte (chapitre ou annexe)
- [x] Titres courts pour les tables des figures et des tableaux

- ☑ Création de la fonction `#subfigure()` pour les sous-figures via le package `subpar`
- ☑ Référencement automatique des sous-figures via la modification de la fonction Typst `#ref()`
- ☑ Recréation des liens hypertextes pour les sous-figures pour la navigation dans le document via la fonction Typst `#link()`

Équations

- ☑ Adaptation de la numérotation des équations en fonction du contexte (chapitre ou annexe)
- ☑ Création d'une fonction pour encadrer les équations importante – `#boxeq()`
- ☑ Création d'une fonction définir des équations sans numérotation – `#nonumeq()`
- ☑ Utilisation du package `equate` pour numéroter des équations d'un système de type (1.1a)

Bibliographie

- ☑ Vérification de la liste des références via `bibtex`
- ☑ Idem pour `hayagriva` (voir documentation)

Bibliographie

[1] L. Mädje, « A Programmable Markup Language for Typesetting », M.S. Thesis. Technische Universität Berlin, 2022.

[2] M. Haug, « Fast Typesetting with Incremental Compilation », M.S. Thesis. Technische Universität Berlin, 2022.