Collection de gabarits Cnam

Rédaction de mémoire en Typst

03-11-2024 Template 0.1.0 Typst 0.12.0

Mathieu AUCEJO

Ce package rassemble une collection de gabarits Typst utilisant la charte graphique conçue par la direction de la Communication du Conservatoire national des arts et métiers.

Table des matières

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) ;

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

- un système de paquets simple d'utilisation (pour rechercher ou voir la liste des paquets, n'hésitez pas à visiter Typst: Universe) ;
- des extensions pour VS Code existent, comme `Typst LSP` ou `Tinymist`, pour avoir des fonctionnalités similaires à `LaTeX Workshop`.

Pour finir, 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 des gabarits

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/cnam_templates.typ": *
   ```

   > L'adresse du fichier à utiliser dans votre fichier principal `nom_de_mon_document.typ` dépend de l'emplacement du dossier `cnam-templates` 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 `cnam-templates` 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 `cnam-templates` (actuellement `cnam-templates/0.1.0`).

```
// Si installation dans le dossier `/typst/packages/local`
#import "@local/cnam_templates:0.1.0": *

// Si installation dans le dossier /typst/packages/preview`
#import "@preview/cnam_templates:0.1.0": *
```

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 des gabarits

Le dossier `cnam_templates` contient actuellement plusieurs gabarits différents, pour rédiger :

- des lettres ;
- des conventions ;
- des documents de réunion ;
- des rapports ;
- des diaporamas.

> Pour utiliser les différents gabarits, il faut que les polices `Raleway` et `Crimson Pro` soient installées sur votre ordinateur.

3.1. Lettres

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

```
#show cnam-lettre.with(
  ...
)
```

3. Utilisation des gabarits

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

```
#cnam-lettre(
  ⟨composante⟩: "cnam",
  ⟨type⟩: none,
  ⟨surtitre⟩: none,
  ⟨destinataire⟩: none,
  ⟨expediteur⟩: none,
  ⟨objet⟩: none,
  ⟨lieu⟩: none,
  ⟨date⟩: none,
  ⟨signature⟩: none
)[⟨body⟩]
```

Argument

⟨composante⟩: "cnam" `str`

Nom de la composante Cnam émettrice de la lettre. Ce paramètre permet de définir le logo qui sera affiché dans l'en-tête de la lettre.

Pour connaître l'ensemble des valeurs possibles, se référer à la section 4.5.1.

Argument

⟨type⟩: none `str`

Type de lettre.

Si l'argument `surtitre` est renseigné, sa valeur remplace celle définie par `type`.

Les valeurs possibles sont :

- `"lettre-officielle"`;
- `"courrier-interne"`;
- `"note-service"`;
- `"note-cadrage"`.

Argument

⟨surtitre⟩: none `array`

Surtitre de la lettre.

Le surtitre permet de personnaliser le type de lettre. Il remplace la valeur définie par `type`.

Exemple : `surtitre: ("Mon", "En-tête")`.

Argument

⟨destinataire⟩: none `dictionary`

Coordonnées du destinataire de la lettre.

Le paramètre `destinataire` est un dictionnaire contenant les clés suivantes :

- `nom` : nom du destinataire ;
- `adresse` : adresse du destinataire.

> Les valeurs associées aux clés sont toute de type `str` ou `content`.

Argument

⟨expediteur⟩: none `dictionary`

Coordonnées de l'expéditeur de la lettre.

Le paramètre `expediteur` est un dictionnaire contenant les clés suivantes :

- `nom` : nom de l'expéditeur ;
- `adresse` : adresse de l'expéditeur ;
- `telephone` : numéro de téléphone de l'expéditeur ;
- `mail` : adresse mail de l'expéditeur.

> Les valeurs associées aux clés sont toute de type `str` ou `content`.

Argument

⟨objet⟩: none `str` | `content`

Objet de la lettre.

Exemple : `objet: "Objet de la lettre"`.

Argument

⟨lieu⟩: none `str` | `content`

Lieu de rédaction de la lettre.

Exemple : `lieu: "Paris"`.

Argument

⟨date⟩: none `str` | `content`

Date de rédaction de la lettre.

Exemple : `date: "01 janvier 2025"`.

Argument

⟨signature⟩: none `content`

Signature de l'expéditeur.

Exemple : `signature: image("ma-signature.png")`.

4.1. Conventions

Pour utiliser le modèle de conventions, celui-ci doit être initialisé en appliquant une règle d'affichage (show rule) avec la commande `#cnam-convention()` en passant les options nécessaires avec l'instruction `with` dans votre fichier principal `typ` :

```
#show cnam-convention.with(
  ...
)
```

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

```
#cnam-convention(
  ⟨composante⟩: "cnam",
  ⟨convention⟩: none,
  ⟨titre⟩: none,
  ⟨partenaires⟩: none,
  ⟨lieu⟩: none,
  ⟨date⟩: none,
  ⟨toc⟩: false
)[⟨body⟩]
```

Argument

⟨composante⟩: "cnam" — `str`

Nom de la composante Cnam émettrice de la convention. Ce paramètre permet de définir le logo qui sera affiché dans l'en-tête de la convention.

Pour connaître l'ensemble des valeurs possibles, se référer à la section 4.5.1.

Argument

⟨convention⟩: none — `str` | `content`

Type de la convention.

La valeur de l'argument `convention` permet de définir la seconde partie du surtitre du document.

Exemple : `convention: "cadre"`.

Argument

⟨titre⟩: none — `str` | `content`

Titre de la convention.

Exemple : `titre: [Convention cadre entre xxx et yyy]`.

Argument

⟨lieu⟩: none — `str` | `content`

Lieu de signature de la convention.

Exemple : `lieu: "Paris"`.

Argument

⟨date⟩: none — `str` | `content`

Date de signature de la convention.

Exemple : `date: "01 janvier 2025"`.

Argument

⟨toc⟩: false — `bool`

Affichage de la table des matières.

Le paramètre `toc` permet d'afficher ou de masquer la table des matières de la convention.

4.2. Documents de réunion

Pour utiliser les modèles de document de réunion, celui-ci doit être initialisé en appliquant une règle d'affichage (show rule) avec la commande `#cnam-reunion()` en passant les options nécessaires avec l'instruction `with` dans votre fichier principal `typ` :

```
#show cnam-reunion.with(
  ...
)
```

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

```
#cnam-reunion(
  ⟨composante⟩: "cnam",
  ⟨type⟩: "cr",
  ⟨titre⟩: none,
  ⟨redacteur⟩: none,
  ⟨lieu⟩: none,
  ⟨date⟩: none,
  ⟨toc⟩: false
)[⟨body⟩]
```

Argument

`⟨composante⟩: "cnam"` `str`

Nom de la composante Cnam émettrice du compte-rendu. Ce paramètre permet de définir le logo qui sera affiché dans l'en-tête du compte-rendu.

Pour connaître l'ensemble des valeurs possibles, se référer à la section 4.5.1.

Argument

`⟨type⟩: "cr"` `str`

Type de document de réunion.

Ce paramètre définit le surtitre du document. Les valeurs possibles de ce paramètre sont :

- `"cr"` : Compte-rendu de réunion ;
- `"pv"` : Procès-verbal de réunion ;
- `"odj"` : Ordre du jour de réunion.

Argument

`⟨titre⟩: none` `str` | `content`

Intitulé de la réunion.

Exemple : `titre: "Conseil de perfectionnement de la LP xxx"`.

Argument

⟨redacteur⟩: none — str | content

Nom du rédacteur du document.

Exemple : `redacteur: "Jean Dupont"`.

Argument

⟨lieu⟩: none — str | content

Lieu de la réunion.

Exemple : `lieu: "Salle des conseils"`.

Argument

⟨date⟩: none — str | content

Date de la réunion.

Exemple : `date: "01 janvier 2025"`.

Argument

⟨toc⟩: false — bool

Affichage de la table des matières du document.

4.3. Rapports

Pour utiliser le modèle de rapport, celui-ci doit être initialisé en appliquant une règle d'affichage (show rule) avec la commande `#cnam-rapport()` en passant les options nécessaires avec l'instruction `with` dans votre fichier principal `typ` :

```
#show cnam-rapport.with(
  ...
)
```

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

`#cnam-rapport(⟨config-titre⟩: (:), ⟨toc⟩: true, ⟨sec-number⟩: true) [⟨body⟩]`

Argument

⟨config-titre⟩: (:) — dictionary

Dictionnaire contenant les paramètres de personnalisation du titre du rapport.

Ce dictionnaire contient les clés suivantes :

- `titre` : titre du rapport
  - type : `str` ou `content`
  - valeur par défaut : none ;

- `surtitre` : surtitre du rapport
  - type `array` ;
  - valeur par défaut : `("En-tête", "personnalisable")` ;
- `composante` : composante Cnam rédactrice du rapport (voir section 4.5.1)
  - type : `str` ;
  - valeur par défaut : `"cnam"` ;
- `alignement` : alignement de la boîte colorée du titre
  - type : `str` ;
  - valeur par défaut : `"irreg"` pour un alignement irrégulier ;
  - autre valeur possible : `"center"` pour un alignement centré.
- `logo` : logo du ou des partenaires rédacteur du rapport
  - type : `content` ou `array` ;
  - valeur par défaut : none.

  Exemple : `logo: (image("logo-partenaire1.png"), image("logo-partenaire2.png"))`.

  > Les logos ne sont affichés que lorsque la valeur de la clé `alignement` est `"center"`.

4.4. Diaporamas

Pour utiliser le modèle de présentation, celui-ci doit être initialisé en appliquant une règle d'affichage (show rule) avec la commande `#cnam-presentation()` en passant les options nécessaires avec l'instruction `with` dans votre fichier principal `typ` :

```
#show cnam-presentation.with(
  ...
)
```

Le modèle `#cnam-presentation()` est un gabarit construit sur la base du paquet Touying (voir la documentation du paquet ici). Le modèle possède un certain nombre de paramètres permettant de personnaliser le document. Voici la liste des paramètres disponibles :

`#cnam-presentation(⟨..args⟩, ⟨composante⟩: "cnam", ⟨color-set⟩: "red")[⟨body⟩]`

Argument

`⟨composante⟩: "cnam"` `str`

Nom de la composante Cnam émettrice de la présentation. Ce paramètre permet de définir le logo qui sera affiché dans l'en-tête de la présentation.

Pour connaître l'ensemble des valeurs possibles, se référer à la section 4.5.1.

Argument

⟨color-set⟩: "red" `dictionary`

Nom du thème de couleurs à utiliser pour la présentation.

Les valeurs possibles sont :

- `"red"` : thème dont la couleur principale est le rouge Cnam ;
- `"medium-blue"` : thème dont la couleur principales est le bleu médium ;
- `"light-blue"` : thème dont la couleurprincipale est le bleu clair.

> Le nom des thèmes de couleurs correspond aux noms des couleurs principales de la charte graphique Cnam rappelées en section 4.5.2.

Argument

⟨..args⟩ `array`

Paramètres supplémentaires à passer au modèle de présentation.

Ces paramètres permettent de personnaliser la présentation en fonction des besoins de l'utilisateur. Leur utilisation est expliquée dans la documentation du paquet Touying.

Le principal paramètre à passer est `config-info` qui permet de définir les informations de la présentation. Ce paramètre est un `dictionary` contenant les clés suivantes :

- `title` : titre de la présentation
  - type : `str` ou `content`.
- `subtitle` : sous-titre de la présentation ;
  - type : `str` ou `content`.
- `over-title` : surtitre de la présentation
  - type : `array`.
- `facade` : type d'illustration sur la diapositive de titre.
  - type : `str` ;
  - valeurs possibles : `"image"` ou `"photo"`.
    - `image` : image d'illustration de la façade du Cnam ;
    - `photo` : photographie de la façade du Cnam.

4.5. Paramètres communs aux gabarits

Cette section présente les paramètres communs à l'ensemble des gabarits de la collection. Ces paramètres permettent de personnaliser les gabarits en fonction des besoins de l'utilisateur.

4.5.1. Paramètre `composante`

La liste des valeurs possibles du paramètre `composante` sont :

Composantes globales

- `"cnam"` : Établissement public
- `"eicnam"` : École d'ingénieurs du Cnam

4. Comment contribuer ?

Centres Cnam Régionaux

- `"ccr-aura"` : Auvergne-Rhône-Alpes
- `"ccr-bfc"` : Bourgogne-Franche-Comté
- `"ccr-bretagne"` : Bretagne
- `"ccr-cvl"` : Centre-Val de Loire
- `"ccr-ge"` : Grand-Est
- `"ccr-guadeloupe"` : Guadeloupe
- `"ccr-hdf"` : Hauts-de-France
- `"ccr-idf"` : Île-de-France
- `"ccr-martinique"` : Martinique
- `"ccr-na"` : Nouvelle-Aquitaine
- `"ccr-normandie"` : Normandie
- `"ccr-occitanie"` : Occitanie
- `"ccr-paca"` : Provence-Alpes-Côte d'Azur
- `"ccr-pdl"` : Pays de la Loire
- `"ccr-polynesie"` : Polynésie
- `"ccr-reunion"` : Réunion
- `"cep"` : Centre Cnam Paris

Centre Cnam à l'étranger

- `"chine"`
- `"liban"`
- `"madagascar"`
- `"maroc"`

Laboratoires et instituts de recherche

- `"cedric"`
- `"ceet"`
- `"crtd"`
- `"dicen"`
- `"dynfluid"`
- `"eren"`
- `"esdr3c"`
- `"esycom"`
- `"foap"`
- `"gbcm"`
- `"gef"`
- `"ht2s"`
- `"iat"`
- `"lafset"`
- `"lcm"`
- `"lifse"`
- `"lirsa"`
- `"lise"`
- `"lmssc"`

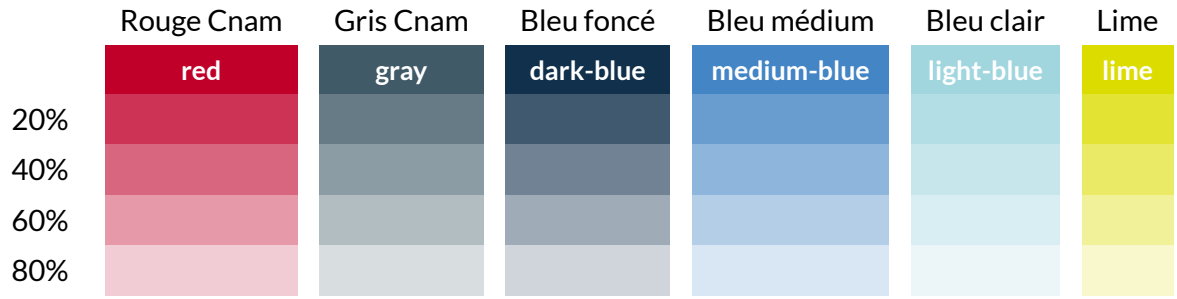
- `"lussac"`
- `"m2n"`
- `"metabiot"`
- `"pimm"`
- `"satie"`
- `"sayfood"`

4.5.2. Couleurs

Les couleurs principales et secondaires de la charte graphique sont accessibles via les différents gabarits.

Couleurs principales

Les couleurs principales sont accessibles via la commande `primary.id_couleur`. De même, les différentes nuanances définies dans la charte graphique peuvent être utilisées via la commande `primary.id_couleur.lighten(value)` Les différentes valeurs des paramètres `id_couleur` et `value` sont :



Couleurs secondaires

Les couleurs principales sont accessibles via la commande `secondary.id_couleur`. Les différentes valeurs des paramètres `id_couleur` disponibles sont :

btp	Bâtiment Travaux publics	**bioagro**	Biotechnologies Bioinformatique Agroalimentaire
eea	Électronique Électrotechnique Automatique	**energie**	Énergie Environnement Développement durable
hygiene	Hygiène & Sécurité Gestion des risques	**chimie**	Industrie & Analyse chimique Cosmétique & Pharma
info	Informatique, Télécoms Médias numériques Cybersécurité	**mater**	Matériaux

4. Comment contribuer ?

meca	Mécanique Acoustique Aérodynamique
metro	Mesure Analyse Contrôle qualité
maths	Modélisation Ingénierie mathématiques Statistiques
ergo	Accompagnement pro Intervention sociale Ergonomie
abf	Assurance, Banque Finance
cmv	Commerce, Marketing Vente
cdcm	Communication Documentation Culture, Médiation
cca	Comptabilité Contrôle Audit
dcd	Droit Criminologie Défense
epidemio	Épidémiologie Addictologie Santé publique
fco	Formation continue Orientation Insertion professionnelle
gpc	Gestion publique et des collectivités
immo	Immobilier
innov	Innovation Développement Prospective, Entrepreneuriat
strategie	Management, Stratégie Organisation d'entreprises, Gestion de projet
mg4s	Management et Gestion des secteurs de santé, sanitaires et sociaux
psycho	Psychologie du travail Psychanalyse
socio	Sociologie du travail
tourisme	Tourisme
trans	Transport & Logistique

ril

Relations internationales
UE & Langues

4. Comment contribuer ?

Si vous souhaitez contribuer à l'amélioration des gabarits ou solliciter la création de nouveaux gabarits, vous pouvez soit :

- ouvrir une *issue* sur le dépôt Github du projet ;
- proposer une *pull request* sur le dépôt, si vous souhaitez ajouter des fonctionnalités, corriger des erreurs ou proposer de nouveaux gabarits ;
- contacter l'auteur par mail à l'adresse suivante : mathieu.aucejo@lecnam.net.

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.