

La classe `tutodoc` - Documentation de type tutoriel

Christophe BAL

25 oct. 2024 - Version 1.6.0

Résumé

La classe `tutodoc`¹ est utilisée par son auteur pour produire de façon sémantique des documentations de packages et de classes L^AT_EX dans un style de type tutoriel² via un rendu sobre pour une lecture sur écran.

Remarque : cette documentation est aussi disponible en anglais.

Abstract.

The `tutodoc` class³ is used by its author to semantically produce documentation of L^AT_EX packages and classes in a tutorial style⁴ using a sober rendering for reading on screen.

Remark : this documentation is also available in French.

Note (Derniers changements).

BIFURCATION.

- L'environnement `showcase` et ses descendants : la clé `color` a été renommée `colstripe`.
- La macro `\tdoclinkcolor` devient la couleur `tutodoc@link@color` destinée à un usage interne.

NOUVEAU.

- L'option de classe `theme` permet de choisir différents thèmes de mise en forme.
- Journal des modifications : ajout de l'environnement `tdoctech` pour les informations techniques.
- L'environnement `showcase` et ses descendants : la clé `coltext` permet de changer aussi la couleur du texte.
- Les nouvelles fonctionnalités ont été documentées.

MISE À JOUR.

- Journal des modifications : pour l'environnement `tdocupdate`, la icône  remplace l'ancienne icône .

1. Le nom vient de « *tuto·rial·type doc·umentation* » qui se traduit en « *documentation de type tutoriel* ».

2. L'idée est de produire un fichier PDF efficace à parcourir pour des besoins ponctuels. C'est généralement ce que l'on attend d'une documentation liée au codage.

3. The name comes from « *tuto·rial·type doc·umentation* ».

4. The idea is to produce an efficient PDF file that can be browsed for one-off needs. This is generally what is expected of coding documentation.

Table des matières

I.	Mises en forme générales imposées	3
	1. Géométrie de la page	3
	2. Titre et table des matières	3
	3. Liens dynamiques	4
II.	Quelle langue est utilisée par la classe <code>tutodoc</code> ?	4
III.	Cela veut dire quoi en « <i>anglais</i> »	4
IV.	Choisir son thème	5
V.	Mettre en avant du contenu	5
	1. Du contenu dans le flot de la lecture	5
	i. Des exemples	5
	ii. Des remarques	6
	2. Du contenu tape-à-l'oeil	6
	i. Une astuce	6
	ii. Note informative	7
	iii. Un truc important	7
	iv. Avertir d'un point très délicat	7
	v. Avertir d'un danger	7
VI.	Indiquer des packages, des classes, des macros ou des environnements	8
VII.	Origine d'un préfixe ou d'un suffixe	8
VIII.	Un rendu en situation réelle	8
	1. Avec une bande colorée	8
	2. Sans bande colorée	10
	3. En important le code \LaTeX	10
IX.	Cas d'utilisation en \LaTeX	11
	1. Codes « <i>en ligne</i> »	11
	2. Codes tapés directement	11
	3. Codes importés	12
	4. Codes importés et mis en situation	13
X.	Indiquer les changements	14
	1. À quel moment ?	14
	2. Quoi de neuf ?	15
XI.	Décorations	16
XII.	Contribuer	17
	1. Compléter les traductions	17
	i. Les dossiers <code>fr</code> et <code>en</code>	17
	ii. Le dossier <code>changes</code>	17
	iii. Le dossier <code>status</code>	17
	iv. Les fichiers <code>README.md</code> et <code>LICENSE.txt</code>	18
	v. De nouvelles traductions	18
	2. Améliorer le code source	19
XIII.	Historique	19

I. Mises en forme générales imposées

1. Géométrie de la page

Le package `geometry` est chargé avec les réglages suivants.

```
\RequirePackage[
  top          = 2.5cm,
  bottom       = 2.5cm,
  left         = 2.5cm,
  right        = 2.5cm,
  marginparwidth = 2cm,
  marginparsep  = 2mm,
  heightrounded
]{geometry}
```

2. Titre et table des matières

Les packages `titlesec` et `tocbasic` sont réglés comme suit.

```
\RequirePackage[raggedright]{titlesec}

% ...
\ifcsundef{chapter}%
  {}%
  {\renewcommand\thechapter{\Alph{chapter}.}}

\renewcommand\thesection{\Roman{section}.}
\renewcommand\thesubsection{\arabic{subsection}.}
\renewcommand\thesubsubsection{\roman{subsubsection}.}

\titleformat{\paragraph}[hang]%
  {\normalfont\normalsize\bfseries}%
  {\theparagraph}{1em}%
  {}

\titlespacing*{\paragraph}%
  {0pt}%
  {3.25ex plus 1ex minus .2ex}%
  {0.5em}

% Source
% * https://tex.stackexchange.com/a/558025/6880
\DeclareTOCStyleEntries[
  raggedentrytext,
  linefill = \hfill,
  indent   = 0pt,
  dynindent,
  numwidth = 0pt,
  numsep   = 1ex,
  dynnumwidth
]{tocline}{
  chapter,
  section,
  subsection,
  subsubsection,
  paragraph,
  subparagraph
}

\DeclareTOCStyleEntry[indentfollows = chapter]{tocline}{section}
```

3. Liens dynamiques

Le package `hyperref` est importé en coulisse avec les réglages ci-dessous.

```
\newcommand{\tdoclinkcolor}{NavyBlue!85!white}

\hypersetup{
  colorlinks,
  citecolor = \tdoclinkcolor,
  filecolor = \tdoclinkcolor,
  linkcolor = \tdoclinkcolor,
  urlcolor = \tdoclinkcolor
}
```

II. Quelle langue est utilisée par la classe `tutodoc` ?

Cette documentation charge le package `babel` via `\usepackage[french]{babel}`. Dès lors, la classe `tutodoc` repère `fr` comme langue principale utilisée par `babel`.⁵ Comme cette langue fait partie de la liste des langues prises en compte, voir ci-dessous, la classe `tutodoc` produira les effets attendus.

- `en` : anglais.
- `es` : espagnol.
- `fr` : français.

Mise en garde.

Si le choix de la langue principale n'est pas faite dans le préambule, le mécanisme employé échouera avec des effets de bord non voulus (voir l'avertissement qui suit).

Avertissement.

Lorsqu'une langue n'est pas prise en compte par `tutodoc`, un message d'avertissement est émis, et l'anglais est alors choisi comme langue vis-à-vis de `tutodoc`.

Note.

Le mécanisme utilisé devrait être compatible avec le package `polyglossia`.

III. Cela veut dire quoi en « *anglais* »

Penser aux non-anglophones est bien, même si ces derniers se font de plus en plus rares.

Cool et top signifient `\tdocinEN*{cool}` et `\tdocinEN{top}`.

Cool et top signifient « *cool* » et « *top* » en anglais.

La macro `\tdocinEN` et sa version étoilée s'appuient sur `\tdocquote` : par exemple, « *sémantique* » s'obtient via `\tdocquote{sémantique}`.

Note.

Le texte « en anglais » est traduit dans la langue détectée par `tutodoc`.

5. Techniquement, on utilise `\BCPdata{language}` qui renvoie une langue au format court.

IV. Choisir son thème

Pour modifier la mise en forme générale, la classe `tutodoc` propose l'option `theme = <choix>` où `<choix>` peut prendre les valeurs suivantes.

- `bw` : ce thème est de type noir-et-blanc avec certaines nuances de gris.
- `color` : ce thème est coloré, *c'est la valeur par défaut*.
- `dark` : ce thème est sombre, idéal pour se reposer les yeux.
- `draft` : ce thème est juste pour une impression papier à la recherche d'erreurs de contenu pas forcément simples à débusquer devant son écran.

Note.

A la fin de ce document, après l'historique des changements, vous trouverez une galerie de cas d'utilisation de ces différents thèmes : se rendre à l'annexe page ??.

V. Mettre en avant du contenu

Note.

Les environnements présentés dans cette section^a ajoutent un court titre indiquant le type d'informations fournies. Ce court texte sera toujours traduit dans la langue repérée par la classe `tutodoc`.

a. La mise en forme provient du package `keytheorems`.

1. Du contenu dans le flot de la lecture

Important.

Tous les environnements présentés dans cette section partagent le même compteur qui sera remis à zéro dès qu'une section de niveau au moins égale à une `\section` sera ouverte.

i. Des exemples

Des exemples numérotés, si besoin, s'indiquent via `\begin{tdocexa} ... \end{tdocexa}` qui propose un argument optionnel pour ajouter un mini-titre. Voici deux usages possibles.

```
\begin{tdocexa}
  Un exemple...
\end{tdocexa}
```

```
\begin{tdocexa}[Mini titre]
  Utile ?
\end{tdocexa}
```

Exemple V.1. *Un exemple...*

Exemple V.2 (Mini titre). *Utile ?*

Astuce.

Il peut parfois être utile de revenir à la ligne dès le début du contenu. Le code suivant montre comment faire (ce tour de passe-passe reste valable pour l'environnement `tdocrem` présenté juste après). Noter au passage que la numérotation suit celle de l'exemple précédent comme souhaité.

```
\begin{tdocexa}
\leavevmode
\begin{enumerate}
\item Point 1.

\item Point 2.
\end{enumerate}
\end{tdocexa}
```

Exemple V.3.

1. Point 1.
2. Point 2.

ii. Des remarques

Tout se passe via `\begin{tdocrem} ... \end{tdocrem}` avec un fonctionnement identique à l'environnement `tdocexa` comme le montre l'exemple suivant.

```
\begin{tdocrem}
  Juste une remarque...
\end{tdocrem}

\begin{tdocrem}[Mini titre]
  Utile ?
\end{tdocrem}
```


Remarque V.4. *Juste une remarque...*

Remarque V.5 (Mini titre). *Utile ?*

2. Du contenu tape-à-l'oeil

Note.

Les icônes sont obtenues via le package `fontawesome5`, et la gestion de l'espacement avec le texte est faite par la macro `\tdocicon`.^a

^a. Par exemple, `\fbox{\tdocicon{\faBed}{Fatigué}}` produit  Fatigué .

i. Une astuce


L'environnement `tdoctip` sert à donner des astuces. Voici comment l'employer.

```
\begin{tdoctip}
  Une astuce.
\end{tdoctip}

\begin{tdoctip}[Mini titre]
  Utile ?
\end{tdoctip}
```

 Astuce.

Une astuce.

 Astuce (Mini titre).

Utile ?

Note.

Les couleurs sont obtenues via les macros développables `\tdocbackcolor` et `\tdocdarkcolor`. Pour des informations complémentaires à ce sujet, se reporter à la fin de la section 1. page 8.

Astuce.

Quelque fois, un contenu mis en avant peut se réduire à une liste. Dans ce cas, la mise en forme peut être améliorée comme suit où nous utilisons l'option `wide` du package `enumitem`.

```

\begin{tdoctrp}[Peu élgant]
  \begin{enumerate}
    \item Point 1.
    \item Point 2.
  \end{enumerate}
\end{tdoctrp}
VERSUS.
\begin{tdoctrp}[Plus élgant]
  \begin{enumerate}[wide]
    \item Point 1.
    \item Point 2.
  \end{enumerate}
\end{tdoctrp}

```

 Astuce (Peu élgant).

1. *Point 1.*
2. *Point 2.*

VERSUS.

 Astuce (Plus élgant).

1. *Point 1.*
2. *Point 2.*

ii. Note informative

L'environnement `tdocnote` sert à mettre en avant des informations utiles. Voici comment l'utiliser.

```

\begin{tdocnote}
  Un truc utile à vous dire...
\end{tdocnote}

\begin{tdocnote}[Mini titre]
  Utile ?
\end{tdocnote}

```

 Note.

Un truc utile à vous dire...

 Note (Mini titre).

Utile ?

iii. Un truc important

L'environnement `tdocimp` permet d'indiquer quelque chose d'important mais sans danger.

```

\begin{tdocimp}
  Un truc important sans danger.
\end{tdocimp}

\begin{tdocimp}[Mini titre]
  Utile ?
\end{tdocimp}

```

 Important.

Un truc important sans danger.

 Important (Mini titre).

Utile ?

iv. Avertir d'un point très délicat

L'environnement `tdoccaut` sert à indiquer un point délicat à l'utilisateur. Voici comment l'employer.

```

\begin{tdoccaut}
  Prudence, prudence...
\end{tdoccaut}

\begin{tdoccaut}[Mini titre]
  Utile ?
\end{tdoccaut}

```

 Mise en garde.

Prudence, prudence...

 Mise en garde (Mini titre).

Utile ?

v. Avertir d'un danger

L'environnement `tdocwarn` sert à avertir l'utilisateur d'un piège à éviter. Voici comment l'employer.

<pre>\begin{tdocwarn} Evitez les dangers... \end{tdocwarn}</pre>	<div>⚠ Avertissement.</div> <div><i>Evitez les dangers...</i></div>
<pre>\begin{tdocwarn}[Mini titre] Utile ? \end{tdocwarn}</pre>	<div>⚠ Avertissement (Mini titre).</div> <div><i>Utile ?</i></div>

VI. Indiquer des packages, des classes, des macros ou des environnements

Voici ce qu'il est possible de taper de façon sémantique.

<code>\tdoccls{maclasse}</code> sert à...	<code>\\</code>	maclasse sert à...
<code>\tdocpack{monpackage}</code> est pour...	<code>\\</code>	monpackage est pour...
<code>\tdocmacro{unemacro}</code> permet de...	<code>\\</code>	<code>\unemacro</code> permet de...
<code>\tdocenv{env}</code> produit...	<code>\\</code>	<code>\begin{env} ... \end{env}</code> produit...
<code>\tdocenv[[opt1]<opt2>]{env}</code>	<code>\\</code>	<code>\begin{env}[opt1]<opt2> ... \end{env}</code>
Juste <code>\tdocenv*{env}</code> ...	<code>\\</code>	Juste env...
Enfin <code>\tdocenv*[[opt1]<opt2>]{env}</code> ...		Enfin env...

Remarque VI.1. Contrairement à `\tdocinlatex`, les macros `\tdocenv` et `\tdocenv*` ne colorent pas le texte produit. De plus, `\tdocenv{monenv}` produit `\begin{monenv} ... \end{monenv}` avec des espaces afin d'autoriser des retours à la ligne si besoin.

⚠ Avertissement.
<i>L'argument optionnel de la macro <code>\tdocenv</code> est copié-collé^a lors du rendu. Ceci peut donc parfois nécessiter d'utiliser des accolades protectrices comme dans l'exemple ci-dessus.</i>
<small>a. Se souvenir que tout est possible ou presque dorénavant.</small>

VII. Origine d'un préfixe ou d'un suffixe

Pour expliquer les noms retenus, rien de tel que d'indiquer et expliciter les courts préfixes et suffixes employés. Ceci se fait facilement comme suit.

<code>\tdocpre{sup}</code> est relatif à...	<code>\\</code>	sup est relatif à...
<code>\tdocprewhy{sup.erbe}</code> signifie...	<code>\\</code>	sup·erbe signifie...
<code>\emph{\tdocprewhy{sup.er} pour...}</code>		<i>sup·er pour...</i>

Remarque VII.1. Le choix du point pour scinder un mot permet d'utiliser des mots avec un tiret comme dans `\tdocprewhy{ca.sse-brique}` qui donne *ca·sse-brique*.

VIII. Un rendu en situation réelle

Il est parfois utile d'obtenir directement le rendu d'un code dans la documentation. Ceci nécessite que ce type de rendu soit dissociable du texte donnant des explications.

1. Avec une bande colorée

Exemple VIII.1 (Avec les textes par défaut). Il peut être utile de montrer un rendu réel directement dans un document.⁶ Ceci se tape via `\begin{tdocshowcase} ... \end{tdocshowcase}` comme suit.

6. Typiquement lorsque l'on fait une démo.


```

\begin{tdocshowcase}
  \bfseries Un peu de code \LaTeX.

  \bigskip

  \emph{\large Fin de l'affreuse démo.}
\end{tdocshowcase}

```

On obtient alors le rendu suivant.⁷

————— Début du rendu réel —————

Un peu de code **LaTeX**.

Fin de l'affreuse démo.

————— Fin du rendu réel —————

Remarque VIII.2. Voir la section 4. page 13 pour obtenir facilement un code suivi de son rendu réel comme dans l'exemple précédent.

i Note.

Les textes explicatifs s'adaptent à la langue détectée par *tutodoc*.

Exemple VIII.3 (Changer les couleurs et/ou les textes par défaut).

```

\begin{tdocshowcase}[before = Mon début,
                        after   = Ma fin à moi,
                        colstripe = red,
                        coltext  = orange!75!black]
  Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}

```

Ceci produira ce qui suit.

————— Mon début —————

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

————— Ma fin à moi —————

i Note.

Dans l'exemple précédent, le texte utilise bien l'orange assombri proposé. Par contre, le rouge sert de base pour obtenir les couleurs utilisées pour la bande : les transformations utilisées dépendent du thème choisi.^a Il faut également savoir qu'en coulisse, la macro `\tdocruler` est employée.

```
\tdocruler[red]{Un pseudo-titre décoré}
```

————— Un pseudo-titre décoré —————

^a. Par exemple, les thèmes `bw` et `draft` ne tiennent pas compte de la clé `colstripe` !

⚠ Avertissement.

Avec les réglages par défaut, si le code *LaTeX* à mettre en forme commence par un crochet ouvrant, il faudra user de `\string` comme dans l'exemple suivant.

7. En coulisse, la bande est créée sans effort grâce au package `clstrip`.

```
\begin{tdocshowcase}
  \string[Cela fonctionne...]
\end{tdocshowcase}
```

Ceci produira ce qui suit.

Début du rendu réel

[Cela fonctionne...]

Fin du rendu réel

2. Sans bande colorée

Le rendu de `\begin{tdocshowcase} ... \end{tdocshowcase}` avec une bande colorée peut ne pas convenir, ou parfois ne pas être acceptable malgré le travail fait par `clrstrip`. Il est possible de ne pas utiliser une bande colorée comme nous allons le voir tout de suite.

Exemple VIII.4. L'emploi de `\begin{tdocshowcase}[nostripe] ... \end{tdocshowcase}` demande de ne pas faire appel à `clrstrip`. Voici un exemple d'utilisation.

```
\begin{tdocshowcase}[nostripe]
  Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Ceci produira ce qui suit.

Début du rendu réel

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

Fin du rendu réel

Exemple VIII.5 (Changer les couleurs et/ou les textes par défaut).

```
\begin{tdocshowcase}[nostripe,
  before = Mon début,
  after   = Ma fin à moi,
  colstripe = green,
  coltext  = purple]
  Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

Ceci produira l'horreur suivante.

Mon début

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

Ma fin à moi

3. En important le code L^AT_EX

Pour obtenir des rendus en important le code depuis un fichier externe, au lieu de le taper, il suffit d'employer la macro `\tdocshowcaseinput` dont l'option reprend la syntaxe de celle de `\begin{tdocshowcase} ... \end{tdocshowcase}` et l'argument obligatoire correspond au chemin du fichier.

Exemple VIII.6. Ce qui suit a été obtenu via `\tdocshowcaseinput{external.tex}`.

Début du rendu réel

Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

Fin du rendu réel

Quant à `\tdocshowcaseinput[colstripe = red, coltext = orange!75!black]{external.tex}`, ceci produira le changement de couleur observable ci-après.

Début du rendu réel

Blablobli, blablobli, blablobli, blablobli, blablobli, blablobli...

Fin du rendu réel

IX. Cas d'utilisation en L^AT_EX

Documenter un package ou une classe se fait efficacement via des cas d'utilisation montrant à la fois du code et le résultat correspondant.⁸

1. Codes « en ligne »

La macro `\tdocinl9atex` permet de taper du code en ligne via un usage similaire à `\verb` ou bien comme une macro standard (voir la gestion des accolades dans le dernier cas ci-dessous). Voici des exemples d'utilisation.

<pre>1: \tdocinl⁹atex \$a^b = c\$ \\ 2: \tdocinl⁹atex+\tdocinl⁹atex \$a^b = c\$ + \\ 3: \tdocinl⁹atex{\tdocinl⁹atex{ \$a^b = c\$ }}</pre>	<pre>1: \$a^b = c\$ 2: \tdocinl⁹atex \$a^b = c\$ 3: \tdocinl⁹atex{ \$a^b = c\$ }</pre>
---	---

i Note.

La macro `\tdocinl9atex` est utilisable dans une note de pied de page : voir ci-dessous.^a De plus, une couleur de fond est volontairement utilisée pour subtilement faire ressortir les codes `\LaTeX`.

a. `$minted = TOP$` a été tapé `\tdocinl9atex+$minted = TOP$+` dans cette note de bas de page..

2. Codes tapés directement

Exemple IX.1 (Face à face). Via `\begin{tdoclatex}[sbs] ... \end{tdoclatex}`, on affichera un code et son rendu côte à côte. Indiquons que *sbs* est pour « **s**·**i**de **b**·**y** **s**·**i**de » soit « côte à côte » en anglais. Considérons le code suivant.

```
\begin{tdoclatex}[sbs]  
  $A = B + C$  
\end{tdoclatex}
```

Ceci produira ce qui suit.

$A = B + C$	$A = B + C$
-------------	-------------

Exemple IX.2 (À la suite). `\begin{tdoclatex} ... \end{tdoclatex}` produit le résultat suivant qui correspond à l'option par défaut *std*.¹⁰

```
$A = B + C$
```

$$A = B + C$$

Exemple IX.3 (Juste le code). Via `\begin{tdoclatex}[code] ... \end{tdoclatex}`, on aura juste le code comme ci-après.

⁸. La mise en forme des codes se fait via le package `minted`.

⁹. Le nom de la macro `\tdocinl9atex` vient de « *in·line L^AT_EX* » soit « *L^AT_EX en ligne* » en anglais.

¹⁰. *std* fait référence au comportement « *standard* » de `tcolorbox` vis à vis de la librairie `minted`.

```
 $A = B + C$ 
```

⚠ Avertissement.

Avec la mise en forme par défaut, si le code commence par un crochet ouvrant, il faudra indiquer explicitement l'option par défaut. Considérons le code suivant.

```
\begin{tdoclatex}[std]
  [Étrange... Ou pas !]
\end{tdoclatex}
```

Ceci produira ce qui suit.

```
[Étrange... Ou pas !]
```

```
[Étrange... Ou pas !]
```

Une autre méthode consiste à utiliser la primitive `\string`. Considérons le code suivant.

```
\begin{tdoclatex}
  \string[Étrange... Ou pas !]
\end{tdoclatex}
```

Ceci produira ce qui suit.

```
[Étrange... Ou pas !]
```

```
[Étrange... Ou pas !]
```

3. Codes importés

Pour les codes suivants, on considère un fichier de chemin relatif `examples-listing-xyz.tex`, et ayant le contenu suivant.

```
% Juste une démo.
 $x y z = 1$ 
```

La macro `\tdoclatexinput`, présentée ci-dessous, attend le chemin d'un fichier et propose les mêmes options que l'environnement `tdoclatex`.

Exemple IX.4 (Face à face).

```
\tdoclatexinput[sbs]{examples-listing-xyz.tex}
```

Ceci produit la mise en forme suivante.

<pre>% Juste une démo. $x y z = 1$</pre>	$xyz = 1$
---	-----------

Exemple IX.5 (À la suite).

```
\tdoclatexinput{examples-listing-xyz.tex}
```

Ceci produit la mise en forme suivante où l'option employée par défaut est `std`.

```
% Juste une démo.
 $x y z = 1$ 
```

$xyz = 1$

Exemple IX.6 (Juste le code).

```
\tdoclatexinput[code]{examples-listing-xyz.tex}
```

Ceci produit la mise en forme suivante.

```
% Juste une démo.
 $x y z = 1$ 
```

4. Codes importés et mis en situation

Exemple IX.7 (Mise en situation). Ce qui suit s'obtient via `\tdoclatexshow{examples-listing-xyz.tex}`.

----- Début du rendu dans cette doc. -----

```
% Juste une démo.
 $x y z = 1$ 
```

Ceci donne :

----- Début du rendu réel -----

$xyz = 1$

----- Fin du rendu réel -----

----- Fin du rendu dans cette doc. -----

i Note.

Les textes par défaut tiennent compte de la langue détectée par `tutodoc`.

Exemple IX.8 (Changer le texte explicatif). Via la clé `explain`, on peut utiliser un texte personnalisé. Ainsi, `\tdoclatexshow[explain = Voici le rendu réel.]{examples-listing-xyz.tex}` produira ce qui suit.

----- Début du rendu dans cette doc. -----

```
% Juste une démo.
 $x y z = 1$ 
```

Voici le rendu réel.

----- Début du rendu réel -----

$xyz = 1$

----- Fin du rendu réel -----

----- Fin du rendu dans cette doc. -----

Exemple IX.9 (Les options disponibles). En plus du texte explicatif, il est aussi possible d'utiliser toutes les options de l'environnement `tdocshowcase`, voir [VIII](#). page 8. Voici un exemple illustrant ceci.

```
\tdoclatexshow[explain = Ce qui vient est coloré...,
before = Rendu ci-après.,
after = Rendu fini.,
```

```
colstripe = orange,
coltext   = blue!70!black]
{examples-listing-xyz.tex}
```

Ceci va produire ce qui suit.

----- Début du rendu dans cette doc. -----

```
% Juste une démo.
$x y z = 1$
```

Ce qui vient est coloré...

----- Rendu ci-après. -----

$xyz = 1$

----- Rendu fini. -----

----- Fin du rendu dans cette doc. -----

X. Indiquer les changements

Afin de faciliter le suivi d'un projet, il est indispensable de fournir un historique indiquant les changements effectués lors de la publication d'une nouvelle version.

1. À quel moment ?

On peut au choix dater quelque chose, ou bien le versionner, dans ce second cas le numéro de version pourra éventuellement être daté.

Exemple X.1 (Dater des nouveautés). La macro `\tdocdate` permet d'indiquer une date dans la marge comme dans l'exemple suivant.

```
Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

\medskip % ATTENTION ! Ceci évite le chevauchement.

\tdocdate{2023-09-24}

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...

\medskip % ATTENTION ! Ceci évite le chevauchement.

\tdocdate[gray]{2020-05-08}

Bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli...

Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...

Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...
```

Ceci donne :

----- Début du rendu réel -----

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

24/09/2023

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...

08/05/2020

Bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli, bli...

Blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo, blo...

Blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu, blu...

----- Fin du rendu réel -----

Exemple X.2 (Versionner des nouveautés en les datant éventuellement). Associer un numéro de version à une nouveauté se fait via la macro `\tdocversion`, la couleur et la date étant des arguments optionnels.

```
|tdocversion[red]{10.2.0-beta}[2023-12-01]
```

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

```
|bigskip % ATTENTION ! Ceci évite le chevauchement.
```

```
|tdocversion{10.2.0-alpha}
```

Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble,
ble. ble. ble. ble. ble. ble. ble. ble. ble. ble. ble. ble...

Ceci donne :

Début du rendu réel

10.2.0-beta
01/12/2023

Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...

10.2.0-alpha[illegible]

Fin du rendu réel

 Important.

1. Les macros `\tdocdate` et `\tdocversion` nécessitent deux compilations.
2. Comme la langue détectée pour cette documentation est le français, la date dans le rendu final est au format JJ/MM/AAAA alors que dans le code celle-ci devra toujours être saisie au format anglais AAAA-MM-JJ.

⚠ Avertissement.

Seul l'emploi du format numérique YYYY-MM-DD est vérifié,^a et ceci est un choix ! Pourquoi cela ? Tout simplement car dater et versionner des explications devrait se faire de façon semi-automatisée afin d'éviter tout bug humain.

a. Techniquement, vérifier la validité d'une date, via L^AT_EX3, ne présente pas de difficulté.

2. Quoi de neuf?

tutodoc propose la macro `\tdocstartproj` et différents environnements pour indiquer rapidement et clairement ce qui a été fait lors des derniers changements.¹¹

Note.

Concernant les icônes, voir la note au début de la section 2. page 6.

Exemple X.3 (Juste pour la toute première version).

<code>\tdocstartproj{Première version du projet.}</code>	Première version du projet.
--	---


Exemple X.4 (Pour les nouveautés).

11. L'utilisateur n'a pas besoin de tous les détails techniques.

```

\begin{tdocnew}
  \item Info 1...
  \item Info 2...
\end{tdocnew}

```

 **NOUVEAU.**


- Info 1...
- Info 2...

Exemple X.5 (Pour les mises à jour).

```

\begin{tdocupdate}
  \item Info 1...
  \item Info 2...
\end{tdocupdate}

```

 **MISE À JOUR.**


- Info 1...
- Info 2...

Exemple X.6 (Pour les bifurcations).

```

\begin{tdocbreak}
  \item Info 1...
  \item Info 2...
\end{tdocbreak}

```

 **BIFURCATION.**


- Info 1...
- Info 2...

Exemple X.7 (Pour les problèmes).

```

\begin{tdocprob}
  \item Info 1...
  \item Info 2...
\end{tdocprob}

```

 **PROBLÈME.**

- Info 1...
- Info 2...

Exemple X.8 (Pour les réparations).

```

\begin{tdocfix}
  \item Info 1...
  \item Info 2...
\end{tdocfix}

```

 **RÉPARATION.**


- Info 1...
- Info 2...

Exemple X.9 (Informations techniques).

```

\begin{tdoctech}
  \item Info 1...
  \item Info 2...
\end{tdoctech}

```

 **INFORMATION TECHNIQUE.**


- Info 1...
- Info 2...

Exemple X.10 (Thématiques aux choix avec une icône).

```

\begin{tdoctopic}{À cacher}{\faEyeSlash}
% Une icône venant de fontawesome5.
  \item Info 1...
  \item Info 2...
\end{tdoctopic}

```

 **À CACHER.**

- Info 1...
- Info 2...

Exemple X.11 (Thématiques aux choix sans icône).

```

\begin{tdoctopic}{La fin des icônes}
  \item Info 1...
  \item Info 2...
\end{tdoctopic}

```

LA FIN DES ICÔNES.

- Info 1...
- Info 2...

XI. Décorations

Finissons cette documentation avec un petit outil de mise en forme qui rend de grands services.

Bla, bla, bla...

```
\tdocsep % Pratique pour délimiter.
```

Ceci fonctionne avec des énumérations.

```
\begin{itemize}
  \item Point souligné.
\end{itemize}
```

```
\tdocsep % Un comportement uniforme.
```

Ble, ble, ble...

Bla, bla, bla...

Ceci fonctionne avec des énumérations.

- Point souligné.

Ble, ble, ble...

XII. Contribuer

Note.

Nul besoin d'être un codeur pour participer aux traductions, y compris pour celles utiles au fonctionnement de `tutodoc`.

1. Compléter les traductions

Note.

L'auteur de `tutodoc` gère les versions françaises et anglaises des traductions.

Mise en garde.

Même si nous allons expliquer comment traduire les documentations, il semble peu pertinent de le faire, car l'anglais devrait suffire de nos jours.^a

^a. L'existence d'une version française est une simple conséquence de la langue maternelle de l'auteur de `tutodoc`.

Les traductions sont organisées grosso-modo comme dans la figure 1 où seuls les dossiers importants pour les traductions ont été « ouverts ».¹²

Un peu plus bas, la section v. donne les démarches à suivre pour ajouter de nouvelles traductions.

i. Les dossiers fr et en

Ces deux dossiers, gérés par l'auteur de `tutodoc`, ont la même organisation ; ils contiennent des fichiers faciles à traduire même si l'on n'est pas un codeur.

ii. Le dossier changes

Ce dossier est un outil de communication où sont indiqués les changements importants sans s'attarder sur les modifications mineures propres à une ou plusieurs traductions.

iii. Le dossier status

Ce dossier permet de savoir où en sont les traductions du point de vue du projet. Tout se passe via des fichiers `YAML` bien commentés, et lisibles par un non codeur.

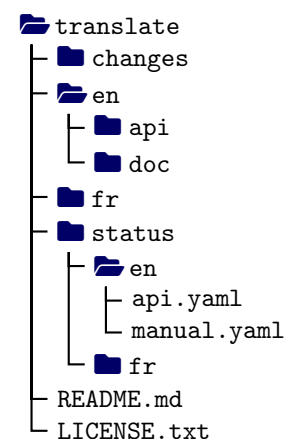


FIGURE 1 – Vue simplifiée du dossier des traductions

¹². Cette organisation était celle du 5 octobre 2024, mais elle reste d'actualité.

iv. Les fichiers README.md et LICENSE.txt

Le fichier LICENSE.txt est bien nommé, tandis que le fichier README.md reprend en anglais les points importants de ce qui est dit dans cette section sur les nouvelles traductions.

v. De nouvelles traductions

Important.

*Le dossier **api** contient les traductions relatives aux fonctionnalités de **tutodoc**. Vous y trouverez des fichiers de type TXT à modifier via un éditeur de texte, ou de code, mais non via un traitement de texte. Les contenus de ces fichiers utilisent des lignes commentées en anglais pour expliquer ce que fera **tutodoc**; ces lignes commencent par `//`. Voici un extrait de ce type de fichier où **les traductions se font après chaque signe = sans toucher à ce qui se trouve avant**, car ce morceau initial est utilisé en interne par le code de **tutodoc**.*

```
// #1: year in format YYYY like 2023.
// #2: month in format MM like 04.
// #3: day in format DD like 29.
date = #1-#2-#3

// #1: the idea is to produce one text like
// "this word means #1 in english".
in_EN = #1 in english
```

Note.

*Le dossier **doc** est réservé aux documentations. Il contient des fichiers de type TEX compilables directement pour une validation en temps réel des traductions faites.*

Avertissement.

*Ne partir que de l'un des dossiers **fr** et **en**, car ceux-ci sont de la responsabilité de l'auteur de **tutodoc**.*

*Imaginons que vous souhaitiez ajouter le support de l'italien en partant de fichiers rédigés en anglais.*¹³

Méthode 1 : git.

1. Obtenir tout le dossier du projet via <https://github.com/bc-tools/for-latex/tree/tutodoc>. Ne pas passer via la branche **main** qui sert à figer les dernières versions stables des projets du dépôt unique <https://github.com/bc-tools/for-latex>.
2. Dans le dossier **tutodoc/contrib/translate**, créer une copie **it** du dossier **en**, le nom court de la langue étant documenté dans la page « *IIETF language tag* » de Wikipédia.
3. Une fois la traduction achevée dans le dossier **it**, il faudra la communiquer via <https://github.com/bc-tools/for-latex/tree/tutodoc> en usant d'un classique **git push**.

Méthode 2 : communiquer par courriels.

1. Via un courriel de sujet « *tutodoc - CONTRIB - en FOR italian* », demander une version des traductions anglaises (noter l'emploi du nom anglais de la nouvelle langue). Bien respecter le sujet du courriel, car l'auteur de **tutodoc** automatise le pré-traitement de ce type de courriels.
2. Vous recevrez un dossier nommé **italian** contenant la version anglaise des dernières traductions. Ce dossier sera le lieu de votre contribution.

¹³. Comme indiqué plus haut, il n'y a pas de besoin réel du côté du dossier **doc** de la documentation.

- Une fois la traduction achevée, il faudra compresser votre dossier `italian` au format `zip` ou `rar` avant de l'envoyer par courriel avec le sujet « *tutodoc - CONTRIB - italian* ».

2. Améliorer le code source

Important.

Si vous souhaitez participer à `tutodoc`, il faudra privilégier le paradigme de programmation \LaTeX 3.

La participation en tant que codeuse, ou codeur, se fait via le dépôt <https://github.com/bc-tools/for-latex/tree/tutodoc> correspondant à la branche de développement `tutodoc`. Ne pas passer via la branche `main` qui sert à figer les dernières versions stables des projets du dépôt unique <https://github.com/bc-tools/for-latex>.

XIII. Historique

1.6.0
25/10/2024



BIFURCATION.

- L'environnement `showcase` et ses descendants : la clé `color` a été renommée `colstripe`.
- La macro `\tdoclinkcolor` devient la couleur `tutodoc@link@color` destinée à un usage interne.

NOUVEAU.

- L'option de classe `theme` permet de choisir différents thèmes de mise en forme.
- Journal des modifications : ajout de l'environnement `tdoctech` pour les informations techniques.
- L'environnement `showcase` et ses descendants : la clé `coltext` permet de changer aussi la couleur du texte.
- Les nouvelles fonctionnalités ont été documentées.

MISE À JOUR.

- Journal des modifications : pour l'environnement `tdocupdate`, la icône  remplace l'ancienne icône .

1.5.0
19/10/2024

INFORMATION TECHNIQUE.

- La version 3 de `minted` est prise en compte.

BIFURCATION.

- La classe `tutodoc` remplace le défunt package `tutodoc` (pour le moment, la toute jeune classe ne propose aucune option spécifique).
- La macro `\tdocruler` s'emploie dorénavant via `\tdocruler[<color>]{<text>}` (avant il fallait utiliser `\tdocruler{<text>}{<color>}`).

NOUVEAU.

- La classe est utilisable en langue espagnole.
- La documentation contient une nouvelle section expliquant comment contribuer.

RÉPARATION.

- La macro `\tdocdate` ne gérait pas le format et la mise en forme de la date.
- Les cadres colorés ne coloraient pas le texte après un saut de page.

1.4.0
28/09/2024

BIFURCATION.

- L'environnement `tdoccaution` a été renommé `tdoccaut` pour une saisie simplifiée.
- Mise en avant de contenus : les exemples et remarques, indiqués via les environnements `tdocexa` et `tdocrem`, sont toujours numérotés via un compteur commun.
- La macro inutilisée `\tdocxspace` a été supprimée.

NOUVEAU.

- Journal des changements : la macro `\tdocstartproj` permet de gérer le cas de la première version publique.
- Factorisation du code : la macro `\tdocicon` est en charge de l'ajout d'icônes devant du texte.

MISE À JOUR.

- Couleurs : les macros `\tdocdarkcolor` et `\tdoclightcolor` proposent un argument facultatif.

1. `\tdocdarkcolor` : la quantité de couleur par rapport au noir peut être définie de manière facultative.
 2. `\tdoclightcolor` : le taux de transparence peut être défini de manière facultative.
- Mise en avant de contenus : réduction de l'espace autour du contenu dans les cadres colorés.
 - Gestion des versions : un meilleur espacement vertical via `\vphantom`.

1.3.1
26/09/2024

◆ NOUVEAU.

- Version étoilée de `\tdocenv` pour n'avoir que le nom de l'environnement.

1.3.0
25/09/2024

🔧 INFORMATION TECHNIQUE.

- La version 3 de `minted` ne peut pas être prise en compte pour le moment car elle comporte des bugs : voir <https://github.com/gpoore/minted/issues/401>. On force donc temporairement l'usage de la version 2 de `minted`.

🔗 BIFURCATION.

- L'environnement `tdocimportant` a été renommé `tdocimp` pour une saisie simplifiée.

◆ NOUVEAU.

- Journal des changements : les environnements proposés utilisent des icônes.
- Mise en avant de contenus : des cadres colorés avec des icônes sont proposés pour les environnements suivants.

- | | | |
|-----------------------------|--------------------------|--------------------------|
| 1. <code>tdoccaution</code> | 3. <code>tdocnote</code> | 5. <code>tdocwarn</code> |
| 2. <code>tdocimp</code> | 4. <code>tdoctip</code> | |

1.2.0-a
23/08/2024

🕒 MISE À JOUR.

- `\tdocversion`
 1. Le numéro de version est au-dessus de la date.
 2. L'espacement est mieux géré lorsque la date est absente.

🔧 RÉPARATION.

- Mise en avant de contenus : les traductions françaises de « *caution* » et « *danger* » étaient erronées.

1.1.0
06/01/2024

◆ NOUVEAU.

- Journal des changements : deux nouveaux environnements.
 1. `\begin{tdocbreak}...\end{tdocbreak}` pour les « *bifurcations* », soit les modifications non rétro-compatibles.
 2. `\begin{tdocprob}...\end{tdocprob}` pour les problèmes repérés.
- `\tdocinlatex` : un jaune léger est utilisé comme couleur de fond.

1.0.1
08/12/2023

🔧 RÉPARATION.

- `\tdocenv` : l'espacement est maintenant correct, même si le paquet `babel` n'est pas chargé avec la langue française.
- `\begin{tdocshowcase}[nostripe]...\end{tdocshowcase}` : les sauts de page autour des lignes « *ca-drantes* » devraient être rares dorénavant.

1.0.0
29/11/2023

🚢 Première version publique du projet.