

La classe `tutodoc` - Documentation de type tutoriel

Christophe BAL

28 sept. 2024 - Version 1.4.0

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.

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

Mises en forme générales imposées	3
Géométrie de la page	3
Titre et table des matières	3
Liens dynamiques	4
Quelle langue est utilisée par la classe <code>tutodoc</code> ?	4
Quel ça veut dire quoi en « <i>anglais</i> »	4
Mettre en avant du contenu	5
Du contenu dans le flot de la lecture	5
Des exemples	5
Des remarques	5
Du contenu tape-à-l'œil	6
Une astuce	6
Note informative	6
Un truc important	6
Avertir d'un point très délicat	7
Avertir d'un danger	7
Indiquer des packages, des classes, des macros ou des environnements	7
Origine d'un préfixe ou d'un suffixe	8
Vu rendu en situation réelle	8
Avec une bande colorée	8
Sans bande colorée	9
En important le code <code>L^AT_EX</code>	10
Chemin d'utilisation en <code>L^AT_EX</code>	10
Codes « <i>en ligne</i> »	11
Codes tapés directement	11
Codes importés	12
Codes importés et mis en situation	13
Indiquer les changements	14
À quel moment ?	14
Quoi de neuf ?	15
Décorations	16
Contribuer	17
Compléter les traductions	17
Les dossiers <code>fr</code> et <code>en</code>	17
Le dossier <code>changes</code>	17
Le dossier <code>status</code>	17
Les fichiers <code>README.md</code> et <code>LICENSE.txt</code>	17
De nouvelles traductions	17
Améliorer le code source	18
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. 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.

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 IV.1. *Un exemple...*

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

Important.

La numérotation des exemples est remise à zéro dès qu'une section de niveau au moins égale à une `\section` est ouverte.

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 IV.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 IV.4. *Juste une remarque...*

Remarque IV.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 `tdoctrp` sert à donner des astuces. Voici comment l'employer.

```

\begin{tdoctrp}
  Une astuce.
\end{tdoctrp}

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

```

 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.

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.

<pre>\begin{tdocimp} Un truc important sans danger. \end{tdocimp}</pre>	 Important. <i>Un truc important sans danger.</i>
<pre>\begin{tdocimp}[Mini titre] Utile ? \end{tdocimp}</pre>	 Important (Mini titre). <i>Utile ?</i>

iv. Avertir d'un point très délicat

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

<pre>\begin{tdoccaut} Prudence, prudence... \end{tdoccaut}</pre>	 Mise en garde. <i>Prudence, prudence...</i>
<pre>\begin{tdoccaut}[Mini titre] Utile ? \end{tdoccaut}</pre>	 Mise en garde (Mini titre). <i>Utile ?</i>

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

V. 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[<i>{opt1}<opt2>}</i>]{env}</code>	<code>\\</code>	<code>\begin{env}[opt1]<opt2> ... \end{env}</code>
Juste <code>\tdocenv*{env}</code> ...	<code>\\</code>	Juste env...
Enfin <code>\tdocenv*[<i>{opt1}<opt2>}</i>]{env}</code> ...		Enfin env...

Remarque V.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.

L'argument optionnel de la macro `\tdocenv` est copié-collé^a lors du rendu. Ceci peut donc parfois nécessiter d'utiliser des accolades protectrices comme dans l'exemple ci-dessus.

a. Se souvenir que tout est possible ou presque dorénavant.

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

Remarque VI.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`.

VII. 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 VII.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.

```
\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 VII.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 VII.3 (Changer la couleur et/ou les textes par défaut).

6. Typiquement lorsque l'on fait une démo.

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


```

\begin{tdocshowcase}[before = Mon début,
                      after  = Ma fin à moi,
                      color  = red]
  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...
————— Ma fin à moi —————

Note.

Vous avez sûrement remarqué que le rouge sert de base pour obtenir les couleurs utilisées.

- La couleur de fond est fournie par `\tdocbackcolor`.
- La couleur des titres et des lignes est fournie par `\tdocdarkcolor`.

Ces macros développables admettent les codes suivants.

```

% Argument 1 : de façon optionnelle, on peut indiquer la quantité de couleur
%               relativement au noir.
%               Il est en général inutile de modifier ce paramètre !
% Argument 2 : une couleur au format xcolor.
\NewExpandableDocumentCommand{\tdocdarkcolor}{0{50}m}{#2!#1!black}

% Argument 1 : de façon optionnelle, on peut indiquer le taux de transparence.
%               Il est en général inutile de modifier ce paramètre !
% Argument 2 : une couleur au format xcolor.
\NewExpandableDocumentCommand{\tdoclightcolor}{0{5}m}{#2!#1}

```

Il faut également savoir qu'en coulisse, la macro `\tdocruler` est utilisée.

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

————— Un pseudo-titre décoré —————

⚠ Avertissement.

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

```

\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 VII.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, bla...
\end{tdocshowcase}
```

Ceci produira ce qui suit.

————— Début du rendu réel —————

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

————— Fin du rendu réel —————

Exemple VII.5 (Changer la couleur et/ou les textes par défaut).

```
\begin{tdocshowcase}[nostripe,
  before = Mon début,
  after  = Ma fin à moi,
  color  = green]
  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 —————

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 VII.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[color = orange]{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 —————

VIII. 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

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

1. Codes « en ligne »

La macro `\tdocinlatex`⁹ 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.

1: <code>\tdocinlatex \$a^b = c\$ </code>	<code>\\</code>	1: <code>\$a^b = c\$</code>
2: <code>\tdocinlatex+\tdocinlatex \$a^b = c\$ +</code>	<code>\\</code>	2: <code>\tdocinlatex \$a^b = c\$ </code>
3: <code>\tdocinlatex{\tdocinlatex{\$a^b = c\$}}</code>		3: <code>\tdocinlatex{\$a^b = c\$}</code>

Note.

La macro `\tdocinlatex` 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 `\tdocinlatex+$minted = TOP$+` dans cette note de bas de page..

2. Codes tapés directement

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

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

Ceci produira ce qui suit.

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

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

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

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

`$A = B + C$`

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

Ceci produira ce qui suit.

9. Le nom de la macro `\tdocinlatex` vient de « *in·line* $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ » soit « $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ en ligne » en anglais.

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

```
[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 VIII.4 (Face à face).

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

Ceci produit la mise en forme suivante.

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

$xyz = 1$

Exemple VIII.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 VIII.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 VIII.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. -----

Note.

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

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

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

$xyz = 1$

IX. 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 IX.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...
```

```
\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 :

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...

Exemple IX.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,
```

Ceci donne :

10.2.0-beta
01/12/2023

10.2.0-alpha


- *Fin du rendu réel*

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


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

15


Exemple IX.6 (Pour les bifurcations).

<pre>\begin{tdocbreak} \item Info 1... \item Info 2... \end{tdocbreak}</pre>	 BIFURCATION. <ul style="list-style-type: none"> • Info 1... • Info 2...
--	--


Exemple IX.7 (Pour les problèmes).

<pre>\begin{tdocprob} \item Info 1... \item Info 2... \end{tdocprob}</pre>	 PROBLÈME. <ul style="list-style-type: none"> • Info 1... • Info 2...
--	---

Exemple IX.8 (Pour les réparations).

<pre>\begin{tdocfix} \item Info 1... \item Info 2... \end{tdocfix}</pre>	 RÉPARATION. <ul style="list-style-type: none"> • Info 1... • Info 2...
--	---

Exemple IX.9 (Thématiques aux choix avec une icône).

<pre>\begin{tdoctopic}{Ne pas ↪ regarder}[\faEyeSlash] % Une icône venant de fontawesome5. \item Info 1... \item Info 2... \end{tdoctopic}</pre>	 NE PAS REGARDER. <ul style="list-style-type: none"> • Info 1... • Info 2...
--	--

Exemple IX.10 (Thématiques aux choix sans icône).

<pre>\begin{tdoctopic}{La fin des icônes} \item Info 1... \item Info 2... \end{tdoctopic}</pre>	LA FIN DES ICÔNES. <ul style="list-style-type: none"> • Info 1... • Info 2...
---	--

X. Décorations

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

<pre>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...</pre>	<p>Bla, bla, bla...</p> <hr/> <p>Ceci fonctionne avec des énumérations.</p> <ul style="list-style-type: none"> • Point souligné. <hr/> <p>Ble, ble, ble...</p>
---	---

XI. 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.

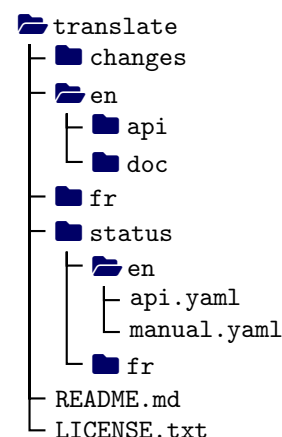
L'auteur de `tutodoc` gère les versions françaises et anglaises des traductions.

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.

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

FIGURE 1 – Vue simplifiée du dossier des traductions

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

fera *tutodoc* ; ces lignes commencent par `//` . Voici un extrait de ce type de fichier.

```
// #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
```

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*.

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.
3. 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

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>.

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

Important.

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

XII. Historique

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, et ils partagent le même compteur.
- 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

PROBLÈME.

- 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 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.