The `tutodoc` class - Tutorial-style documentation

Christophe BAL

Oct 19, 2024 - Version 1.5.0

Abstract

The `tutodoc` class[1] is used by its author to semantically produce documentation of LaTeX packages and classes in a tutorial style[2] using a sober rendering for reading on screen.

Remark : *this documentation is also available in French.*

Note (Derniers changements)**.**

Break.

- *The `tutodoc` class replaces the now-defunct `tutodoc` package (for the moment, the young class offers no specific options).*
- *The `\tdocruler` macro is now used via `tdocruler[<color>]{<text>}` (previously you had to use `tdocruler{<text>}{<color>}`).*

Fix.

- *Version 3 of `minted` is taken into account.*
- *The `\tdocdate` macro did not handle date format and formatting.*
- *Colored frames did not color text after a page break.*

New.

- *The class is usable in Spanish.*
- *The documentation contains a new section explaining how to contribute.*

[1]The name comes from "***tuto·rial-type doc·umentation***".

[2]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.

Contents

I. General formatting imposed

1. Page geometry

The `geometry` package is loaded with the following settings.

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

2. Title and table of contents

The `titlesec` and `tocbasic` packages are set as follows.

```
\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. Dynamic links

The `hyperref` package is imported behind the scenes with the settings below.

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

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

II. What language is used by the `tutodoc` class?

This documentation loads the `babel` package via `\usepackage[english]{babel}`. As a result, the `tutodoc` class identifies `en` as the main language used by `babel`.[3] As this language is included in the list of languages taken into account, see below, the `tutodoc` class will produce the expected effects.

- `en` : English.
- `es` : Spanish.
- `fr` : French.

Caution.

If the choice of main language is not made in the preamble, the mechanism used will fail with unintended side effects (see warning that follows).

Warning.

When a language is not supported by `tutodoc`, a warning message is issued, and English is selected as the language for `tutodoc`.

Note.

The mechanism used should be compatible with the `polyglossia` package.

III. What does that mean in "*English*"?

The macro `\tdocinEN` and its starred version are useless for English speakers because they have the following effects.

```
Cool and top stand for \tdocinEN*{cool} and \tdocinEN{top}.
```

Cool and top stand for "*cool*" and "*top*" in english.

The macro `\tdocinEN` and its starred version are based on `\tdocquote` : for example, "*semantic*" is obtained via `tdocquote{semantic}` .

Note.

*As the text "*in English*" is translated into the language detected by `tutodoc`, the macro `\tdocinEN` and its starred version become useful for non-English speakers.*

[3]Technically, we use `\BCPdata{language}` which returns a language in short format.

IV. Highlighting content

Note.

The environments presented in this section [a] add a short title indicating the type of information provided. This short text will always be translated into the language detected by the `tutodoc` class.

[a]The formatting comes from the `keytheorems` package.

1. Content in the reading flow

Important.

All the environments presented in this section share the same counter.

i. Examples

Numbered examples, if required, are indicated via `\begin{tdocexa} ...\end{tdocexa}`, which offers an optional argument for adding a mini-title. Here are two possible uses.

```
\begin{tdocexa}
    An example...
\end{tdocexa}

\begin{tdocexa}[Mini title]
    Useful?
\end{tdocexa}
```

Example IV.1. *An example...*

Example IV.2 (Mini title)**.** *Useful?*

Important.

The numbering of the examples is reset to zero as soon as a section with a level at least equal to a `\section` is opened.

Tip.

It can sometimes be useful to return to the line at the start of the content. The code below shows how to proceed (this trick also applies to the `tdocrem` environment presented next). Note in passing that the numbering follows that of the previous example as desired.

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

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

Example IV.3.

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

ii. Some remarks

Everything happens via `\begin{tdocrem} ...\end{tdocrem}`, which works identically to the `tdocexa` environment, as shown in the following example.

```
\begin{tdocrem}
    Just one remark...
\end{tdocrem}

\begin{tdocrem}
    Another?
\end{tdocrem}

\begin{tdocrem}[Mini title]
    Useful?
\end{tdocrem}
```

Remark IV.4. *Just one remark...*

Remark IV.5. *Another?*

Remark IV.6 (Mini title)**.** *Useful?*

2. Flashy content

Note.

Icons are obtained via the `fontawesome5` package, and text spacing is managed by the `\tdocicon` macro. [a]

[a] For example, `\fbox{\tdocicon{\faBed}{Tired}}` produces Tired.

i. A tip

The `tdoctip` environment is used to give tips. Here's how to use it.

```
\begin{tdoctip}
    A tip.
\end{tdoctip}

\begin{tdoctip}[Mini title]
    Useful?
\end{tdoctip}
```

Tip.

A tip.

Tip (Mini title)**.**

Useful?

Note.

Colors are obtained via the expandable macros `\tdocbackcolor` and `\tdocdarkcolor`. For further information, please refer to the end of the section 1. page 8.

ii. Informative note

The `tdocnote` environment is used to highlight useful information. Here's how to use it.

```
\begin{tdocnote}
    Something useful to tell you...
\end{tdocnote}

\begin{tdocnote}[Mini title]
    Useful?
\end{tdocnote}
```

Note.

Something useful to tell you...

Note (Mini title)**.**

Useful?

iii. Something important

The `tdocimp` environment is used to indicate something important but harmless.

```
\begin{tdocimp}
    Important and harmless.
\end{tdocimp}

\begin{tdocimp}[Mini title]
    Useful?
\end{tdocimp}
```

Important.

Important and harmless.

Important (Mini title).

Useful?

iv. Caution about a delicate point

The `tdoccaut` environment is used to indicate a delicate point to the user. Here's how to use it.

```
\begin{tdoccaut}
    Caution, caution...
\end{tdoccaut}

\begin{tdoccaut}[Mini title]
    Useful?
\end{tdoccaut}
```

Caution.

Caution, caution...

Caution (Mini title).

Useful?

v. Warning of danger

The `tdocwarn` environment is used to warn the user of a trap to avoid. Here's how to use it.

```
\begin{tdocwarn}
    Avoid the dangers...
\end{tdocwarn}

\begin{tdocwarn}[Mini title]
    Useful?
\end{tdocwarn}
```

Warning.

Avoid the dangers...

Warning (Mini title).

Useful?

V. Specify packages, classes, macros or environments

Here's what you can type semantically.

```
\tdoccls{myclass} is for...                      \\
\tdocpack{mypackage} is for...                   \\
\tdocmacro{onemacro} is for...                   \\
\tdocenv{env} produces...                        \\
\tdocenv[{[opt1]<opt2>}]{env}                    \\
Just \tdocenv*{env}...                           \\
Finally \tdocenv*[{[opt1]<opt2>}]{env}...
```

`myclass` is for...
`mypackage` is for...
`\onemacro` is for...
`\begin{env}` ...`\end{env}` produces...
`\begin{env}[opt1]<opt2>` ...`\end{env}`
Just `env`...
Finally `env`...

Remark V.1. *Unlike `\tdocinlatex`, `\tdocenv` and `\tdocenv*` macros don't color the text they produce. In addition, `\tdocenv{monenv}` produces `\begin{monenv}` ...`\end{monenv}` with spaces to allow line breaks if required.*

Warning.

The optional argument of the `\tdocenv` macro is copied and pasted [a] when rendering. This may sometimes require the use of protective braces, as in the example above.

[a]Remember that almost anything is possible from now on.

VI. Origin of a prefix or suffix

To explain the names chosen, there is nothing like indicating and explaining the short prefixes and suffixes used. This is easily done as follows.

```
\tdocpre{sup} relates to...      \\
\tdocprewhy{sup.erbe} means...   \\
\emph{\tdocprewhy{sup.er} for...}
```

sup relates to...
sup·**erbe** means...
*sup·**er** for...*

Remark VI.1. *The choice of a full stop to split a word allows words with a hyphen to be used, as in `\tdocprewhy{bric.k-breaker}` which gives bric·**k-breaker**.*

VII. A real-life rendering

It is sometimes useful to render code directly in the documentation. This type of rendering must be dissociable from the explanatory text.

1. With a colored stripe

Example VII.1 (With default text)**.** *It can be useful to show a real rendering directly in a document.* [4] *This is done via `\begin{tdocshowcase}`...`\end{tdocshowcase}` as follows.*

```
\begin{tdocshowcase}
    \bfseries A bit of code \LaTeX.

    \bigskip

    \emph{\large End of the awful demo.}
\end{tdocshowcase}
```

The result is the following rendering. [5]

Start of the real output

A bit of code LaTeX.

End of the awful demo.

End of the real output

Remark VII.2. *See the section 4. on page 13 to easily obtain code followed by its actual rendering as in the previous example.*

Note.

The explanatory texts adapt to the language detected by `tutodoc`.

Example VII.3 (Change the default colour and/or text)**.**

```
\begin{tdocshowcase}[before = My beginning,
                     after  = My end,
                     color  = red]
```

[4]Typically when making a demo.
[5]Behind the scenes, the strip is created effortlessly using the `clrstrip` package.

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

This will produce the following.

My beginning

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

My end

Note.

You've probably noticed that red is used as a base to obtain the colors used.

- *The background color is provided by* `\tdocbackcolor`*.*
- *The color of titles and lines is provided by* `\tdocdarkcolor`*.*

These expandable macros accept the following codes.

```
% Argument 1 : optionally, the amount of color relative to black can be specified.
%              In general, there's no need to change this setting!
% Argument 2 : a color in xcolor format.
\NewExpandableDocumentCommand{\tdocdarkcolor}{O{50}m}{#2!#1!black}

% Argument 1 : optionally, the transparency rate can be specified.
%              In general, there's no need to change this setting!
% Argument 2 : a color in xcolor format.
\NewExpandableDocumentCommand{\tdoclightcolor}{O{5}m}{#2!#1}
```

You also have to know that behind the scene, the `\tdocruler` *macro is used.*

```
\tdocruler[red]{A decorated pseudo-title}
```

A decorated pseudo-title

Warning.

With the default settings, if the code to be formatted begins with an opening bracket, use `\string` *as in the following example.*

```
\begin{tdocshowcase}
    \string[This works...]
\end{tdocshowcase}
```

This will produce the following.

Start of the real output

[This works...]

End of the real output

2. Without a colour strip

The rendering of `\begin{tdocshowcase}...\end{tdocshowcase}` with a coloured strip may not be suitable, or sometimes may not be acceptable despite the work done by `clrstrip`. It is possible not to use a coloured strip, as we will see straight away.

Example VII.4. *The use of* `\begin{tdocshowcase}[nostripe] ... \end{tdocshowcase}` *indicate to not use* `clrstrip`. *Here is an example.*

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

This will produce the following.

Start of the real output

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

End of the real output

Example VII.5 (Change the default colour and/or text)**.**

```latex
\begin{tdocshowcase}[nostripe,
                     before = My beginning,
                     after  = My end,
                     color  = green]
    Bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla...
\end{tdocshowcase}
```

This will produce the following.

My beginning

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

My end

3. By importing the LaTeX code

To obtain renderings by importing the code from an external file, instead of typing it, simply use the `\tdocshowcaseinput` macro whose option uses the syntax of that of `\begin{tdocshowcase}` ... `\end{tdocshowcase}` and the mandatory argument corresponds to the path of the file.

Example VII.6. *The following was obtained via* `\tdocshowcaseinput{external.tex}`.

Start of the real output

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

End of the real output

As for `\tdocshowcaseinput[color = orange]{external.tex}` *, this will produce the colour change shown below.*

Start of the real output

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

End of the real output

VIII. Use cases in LaTeX

Documenting a package or class is best done through use cases showing both the code and the corresponding result. [6]

[6] Code is formatted using the `minted` package.

1. “*Inline*” codes

The `\tdocinlatex` macro [7] can be used to type inline code in a similar way to `\verb` or like a standard macro (see brace management in the last case below). Here are some examples.

```
1: \tdocinlatex|$a^b = c$|             \\
2: \tdocinlatex+\tdocinlatex|$a^b = c$|+ \\
3: \tdocinlatex{\tdocinlatex{$a^b = c$}}
```

```
1: $a^b = c$
2: \tdocinlatex|$a^b = c$|
3: \tdocinlatex{$a^b = c$}
```

Note.

The `\tdocinlatex` macro can be used in a footnote: see below. [a] *In addition, a background color is deliberately used to subtly highlight the codes `\LaTeX`.*

[a] `$minted = TOP$` has been typed `\tdocinlatex+$minted = TOP$+` in this footnote...

2. Directly typed codes

Example VIII.1 (Side by side). *Using `\begin{tdoclatex}[sbs]` ... `\end{tdoclatex}`, we can display a code and its rendering side by side. Consider the following code.*

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

This will produce the following.

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

$A = B + C$

Example VIII.2 (Following). *`\begin{tdoclatex}` ... `\end{tdoclatex}` produces the following result, which corresponds to the default option `std`.* [8]

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

$A = B + C$

Example VIII.3 (Just the code). *Via `\begin{tdoclatex}[code]` ... `\end{tdoclatex}`, we'll just get the code as shown below.*

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

Warning.

With default formatting, if the code begins with an opening bracket, the default option must be explicitly indicated. Consider the following code.

```
\begin{tdoclatex}[std]
    [Strange... Or not!]
\end{tdoclatex}
```

This will produce the following.

[7] The name of the macro `\tdocinlatex` comes from “*in·line* LaTeX”.
[8] `std` refers to the “*standard*” behaviour of `tcolorbox` in relation to the `minted` library.

```
[Strange... Or not!]
```

[Strange... Or not!]

Another method is to use the `\string` primitive. Consider the following code.

```
\begin{tdoclatex}
    \string[Strange... Or not!]
\end{tdoclatex}
```

This will produce the following.

```
[Strange... Or not!]
```

[Strange... Or not!]

3. Imported codes

For the following codes, consider a file with the relative path `examples-listing-xyz.tex`, and with the following contents.

```
% Just one demo.
$x y z = 1$
```

The `\tdoclatexinput` macro, shown below, expects the path of a file and offers the same options as the `tdoclatex` environment.

Example VIII.4 (Side by side).

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

This produces the following layout.

```
% Just one demo.
$x y z = 1$
```

$xyz = 1$

Example VIII.5 (Following).

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

This produces the following formatting where the default option is `std`.

```
% Just one demo.
$x y z = 1$
```

$xyz = 1$

Example VIII.6 (Just the code).

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

This produces the following layout.

```
% Just one demo.
$x y z = 1$
```

4. Imported codes put into practice

Example VIII.7 (Showcase). *The following comes from* `\tdoclatexshow{examples-listing-xyz.tex}`.

Start of the rendering in this doc.

```
% Just one demo.
$x y z = 1$
```

This gives :

Start of the real output

$xyz = 1$

End of the real output

End of rendering in this doc.

Note.

The default texts take into account the language detected by `tutodoc`.

Example VIII.8 (Changing the explanatory text). *Using the key* `explain`, *you can use custom text. Thus,* `tdoclatexshow[explain = Here is the actual rendering.]{examples-listing-xyz.tex}` *will produce the following.*

Start of the rendering in this doc.

```
% Just one demo.
$x y z = 1$
```

Here is the actual rendering.

Start of the real output

$xyz = 1$

End of the real output

End of rendering in this doc.

Example VIII.9 (The options available). *In addition to the explanatory text, it is also possible to use all the options of* `tdocshowcase` *environment, see VII. page 8. Here is an example to illustrate this.*

```
\tdoclatexshow[explain = What comes next is colourful...,
               before  = Rendering below.,
               after   = Finished rendering.,
               color   = orange]
               {examples-listing-xyz.tex}
```

This will produce the following.

Start of the rendering in this doc.

```
% Just one demo.
$x y z = 1$
```

What comes next is colourful...

Rendering below.

$xyz = 1$

Finished rendering.

End of rendering in this doc.

IX. Indicate changes

To make it easier to monitor a project, it is essential to provide a history indicating the changes made when a new version is published.

1. When?

You can either date something, or version it, in which case the version number can be dated.

Example IX.1 (Dating new products). *The `\tdocdate` macro is used to indicate a date in the margin, as in the following example.*

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

\medskip % CAUTION! This prevents overlapping.

\tdocdate{2023-09-24}

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

\medskip % CAUTION! This prevents overlapping.

\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...
```

This gives :

Start of the real output

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

2023-09-24 *Ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble, ble...*

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...

End of the real output

Example IX.2 (Versioning new features, possibly with a date). *Associating a version number with a new feature is done using the `\tdocversion` macro, with the colour and date being optional arguments.*

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

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

\bigskip % CAUTION! This prevents overlapping.

\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...
```

This gives :

Start of the real output

10.2.0-beta
2023-12-01
10.2.0-alpha

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

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...

End of the real output

Important.

1. *The `\tdocdate` and `\tdocversion` macros require two compilations.*
2. *The final rendering of the dates takes into account the language detected by `tutodoc`: for example, if French is selected, the dates will be displayed in the format `DD/MM/YYYY`.*

Warning.

Only the use of the digital format `YYYY-MM-DD` is verified, [a] *and this is a choice! Why? Quite simply because dating and versioning explanations should be done semi-automatically to avoid any human bugs.*

[a]Technically, checking the validity of a date using LaTeX3 presents no difficulty.

2. What's new?

`tutodoc` offers the macro `\tdocstartproj` and different environments to indicate quickly and clearly what has been done during the latest changes.[9]

Note.

For icons, see the note at the beginning of the section 2. page 6.

Example IX.3 (Just for the very first version).

```
\tdocstartproj{1st version of the project.}
```

⚓ *1st version of the project.*

Example IX.4 (For new features).

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

New.

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

Example IX.5 (For updates).

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

Update.

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

[9]The user doesn't need all the technical details.

Example IX.6 (For breaks).

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

Break.

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

Example IX.7 (For problems).

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

Problem.

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

Example IX.8 (For fixes).

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

Fix.

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

Example IX.9 (Selectable themes with an icon).

```
\begin{tdoctopic}{Don't look}[\faEyeSlash]
% An icon from fontawesome5.
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

Don't look.

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

Example IX.10 (Selectable themes without icons).

```
\begin{tdoctopic}{End of icons}
    \item Info 1...
    \item Info 2...
\end{tdoctopic}
```

End of icons.

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

X. Ornaments

Let's finish this documentation with a small formatting tool that is very useful.

```
Bla, bla, bla...

\tdocsep % Practical for demarcation.

This works with enumerations.

\begin{itemize}
    \item Underline.
\end{itemize}

\tdocsep % Uniform behaviour.

Ble, ble, ble...
```

Bla, bla, bla...

This works with enumerations.

- Underline.

Ble, ble, ble...

XI. Contribute

Note.

You don't need to be a coder to take part in translations*, including those that are useful for the running of* `tutodoc`*.*

1. Complete the translations

Note.

The author of `tutodoc` *manages the French and English versions of the translations.*

Caution.

Although we're going to explain how to translate the documentation, it doesn't seem relevant to do so, as English should suffice these days.[a]

[a]The existence of a French version is simply a consequence of the native language of the author of `tutodoc`.

The translations are roughly organized as in figure 1 where only the folders important for the translations have been "*opened*".[10] **A little further down, the v. section explains how to add new translations.**

```
translate
├ changes
├ en
│ ├ api
│ └ doc
├ fr
├ status
│ ├ en
│ │ ├ api.yaml
│ │ └ manual.yaml
│ └ fr
├ README.md
└ LICENSE.txt
```

Figure 1: Simplified view of the translation folder

i. The `fr` and `en` folders

These two folders, managed by the author of `tutodoc`, have the same organization; they contain files that are easy to translate even if you're not a coder.

ii. The `changes` folder

This folder is a communication tool where important changes are indicated without dwelling on minor modifications specific to one or more translations.

iii. The `status` folder

This folder is used to keep track of translations from the project's point of view. Everything is done via well-commented `YAML` files, readable by a non-coder.

iv. The `README.md` and `LICENSE.txt` files

The `LICENSE.txt` file is aptly named, while the `README.md` file takes up in English the important points of what is said in this section about new translations.

v. New translations

Important.

The `api` *folder contains translations relating to the functionalities of* `tutodoc`*. Here you'll find* `TXT` *files for editing with a text or code editor, but not with a word processor. The content of these files uses commented lines in English to explain what* `tutodoc` *will do; these lines begin with* `//`*. Here's an extract from such a file, where translations are made after each = sign, without touching the preceding,*

[10]This was the organization on October 5, 2024, but it's still relevant today.

as this initial piece is used internally by the `tutodoc`*code.*

```
// #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.

The `doc` *folder is reserved for documentation. It contains files of type* `TEX` *that can be compiled directly for real-time validation of translations.*

Warning.

Only start from one of the `fr` *and* `en` *folders, as these are the responsibility of the* `tutodoc` *author.*

Let's say you want to add support for Italian from files written in English.[11]

Method 1 : `git`.

1. Obtain the entire project folder via `https://github.com/bc-tools/for-latex/tree/tutodoc` . Do not use the `main` branch, which is used to freeze the latest stable versions of projects in the single `https://github.com/bc-tools/for-latex` repository,.
2. In the `tutodoc/contrib/translate` folder, create a `it` copy of the `en` folder, with the short name of the language documented in the page "*IIETF language tag*" from `Wikipedia`.
3. Once the translation is complete in the `it` folder, you'll need to communicate it via `https://github.com/bc-tools/for-latex/tree/tutodoc` using a classic `git push` .

Method 2 : communicate by e-mail.

1. By e-mail with the subject "`tutodoc - CONTRIB - en FOR italian`", request a version of the English translations (note the use of the English name for the new language). Be sure to respect the subject of the e-mail, as the author of `tutodoc` automates the pre-processing of this type of e-mail.
2. You will receive a folder named `italian` containing the English version of the latest translations. This folder will be the place for your contribution.
3. Once the translation is complete, you will need to compress your `italian` file in `zip` or `rar` format before sending it by e-mail with the subject "`tutodoc - CONTRIB - italian`" .

2. Improving the source code

Important.

If you want to participate in `tutodoc` *you'll need to use the LaTeX3 programming paradigm.*

Participation as a coder is made via the `https://github.com/bc-tools/for-latex/tree/tutodoc` repository corresponding to the `tutodoc` development branch. Do not use the `main` branch, which is used to freeze the latest stable versions of projects in the single `https://github.com/bc-tools/for-latex` repository.

[11] As mentioned above, there is no real need for the `doc` documentation folder.

XII. History

1.5.0
2024-10-19

Break.

- The `tutodoc` class replaces the now-defunct `tutodoc` package (for the moment, the young class offers no specific options).
- The `\tdocruler` macro is now used via `tdocruler[<color>]{<text>}` (remember that the old syntax was `tdocruler{<text>}{<color>}`).

Fix.

- Version 3 of `minted` is taken into account.
- The `\tdocdate` macro did not handle date format and formatting.
- Colored frames did not color text after a page break.

New.

- The class is usable in Spanish.
- The documentation contains a new section explaining how to contribute.

1.4.0
2024-09-28

Break.

- The `tdoccaution` environment has been renamed `tdoccaut` for simplified input.
- Content highlighting: examples and remarks, indicated via the `tdocexa` and `tdocrem` environments, are always numbered using a common counter.
- The unused macro `\tdocxspace` has been deleted.

New.

- Change log: the `\tdocstartproj` macro is used to manage the case of the first public version.
- Code factorization: the `\tdocicon` macro is responsible for adding icons in front of text.

Update.

- Colors: the `\tdocdarkcolor` and `\tdoclightcolor` macros offer an optional argument.
    1. `\tdocdarkcolor` : the amount of color in relation to black can be optionally defined.
    2. `\tdoclightcolor` : the transparency rate can be optionally defined.
- Content highlighting: reduced space around content in colored frames.
- Versioning: better vertical spacing thanks to `\vphantom`.

1.3.1
2024-09-26

New.

- Star version of `\tdocenv` to display only the environment name.

1.3.0
2024-09-25

Problem.

- Version 3 of `minted` cannot be used for the moment as it contains bugs: see https://github.com/gpoore/minted/issues/401. We therefore force the use of version 2 of `minted`.

Break.

- The `tdocimportant` environment has been renamed `tdocimp` for simplified input.

New.

- Change log: proposed environments use icons.
- Content highlighting: colored frames with icons are proposed for the following environments.
    1. `tdoccaution`
    2. `tdocimp`
    3. `tdocnote`
    4. `tdoctip`
    5. `tdocwarn`

1.2.0-a
2024-08-23

Update.

- `\tdocversion`
    1. The version number is above the date.

2. The spacing is better managed when the date is absent.

Fix.

- Content highlighting: the French translations of "*caution*" and "*danger*" were incorrect.

1.1.0
2024-01-06

New.

- Change log : two new environments.
  1. `\begin{tdocbreak}...\end{tdocbreak}` for breaking changes which are not backward compatible.
  2. `\begin{tdocprob}...\end{tdocprob}` for identified problems.
- `\tdocinlatex`: a light yellow is used as the background color.

1.0.1
2023-12-08

Fix.

- `\tdocenv`: spacing is now correct, even if the `babel` package is not loaded with the French language.
- `\begin{tdocshowcase}[nostripe]...\end{tdocshowcase}`: page breaks around "*framing*" lines should be rare from now on.

1.0.0
2023-11-29

First public version of the project.