docshots: LaTeX Package that Renders TeX Samples Next to Their PDF Snapshots*

Yegor Bugayenko
yegor256@gmail.com

2022-10-25, 0.0.5

NB! This package doesn't work on Windows! Also, you must run TeX processor with `--shell-escape` option. Also, you must have `pdlaftex`, Perl, Ghostscript, and pdfcrop installed.

1 Introduction

When you want to demonstrate to the readers of your documentation how to use certain TeX commands, the best way would be to show exactly how the entire document will be rendered in PDF, using a subprocess that would render it (via `pdflatex`, for example). To my best knowledge, there were no packages that would allow you do exactly this. That's why I created this simple package. For example, this code:

```
\begin{docshot}
\documentclass{article}
\usepackage{xcolor}
\pagestyle{empty}
\begin{document}
  Hello, {\color{orange}\LaTeX}!
\end{document}
\end{docshot}
```

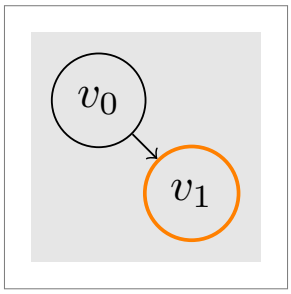
is rendered as such:

Hello, LaTeX!

```
\documentclass{article}
\usepackage{xcolor}
\pagestyle{empty}
\begin{document}
  Hello, {\color{orange}\LaTeX}!
\end{document}
```

Here is a more complex example:

*The sources are in GitHub at yegor256/docshots




```
\documentclass{article}
\usepackage{tikz}
\usetikzlibrary{backgrounds}
\pagestyle{empty}
\begin{document}
\begin{tikzpicture}[
  show background rectangle,
  background rectangle/.style=
    {draw=none,fill=gray!20}]
\node [circle,draw] (v0) {$v_0$};
\node [circle,draw=orange,thick,
  below right of=v0] (v1) {$v_1$};
\draw [->] (v0) -- (v1);
\end{tikzpicture}
\end{document}
```

The picture you see on the left side is rendered by a subprocess executing `pdflatex` with the `.tex` content taken from the source file. After a successful processing of TEX sources, we use pdfcrop to trim the document.

We execute `pdflatex` with `-interaction=batchmode` and `-halt-on-error` options. This means that TEX processing will stop at the first error. Check your TEX log to understand what went wrong.

When you render a text instead of a single picture, it's recommended to use geometry package to change the size of the page and then `empty` page style to remove page numbering:

> "There is no sadder thing than a young pessimist, except an old pessimist" — *Mark Twain*

```
\documentclass{article}
\usepackage[paperwidth=2in]{geometry}
\pagestyle{empty}
\begin{document}
``There is no sadder thing than
a young pessimist, except an old
pessimist'' --- \emph{Mark Twain}
\end{document}
```

2 Package Options

`pdflatex` The default command line tool for turning `.tex` into `.pdf` is `pdflatex`. However, you can change that by using `pdflatex` package option, for example:

```
\documentclass{article}
\usepackage[pdflatex=/usr/local/bin/pdflatex]{docshot}
\begin{document}
\begin{docshot}
Hello, world!
\end{docshot}
\end{document}
```

`gs` The default location of Ghostscript is just `gs`. You can change that by using `gs` package

option, for example:

```
\usepackage[gs=/usr/bin/ghostscript]{docshot}
```

`pdfcrop` The default location of `pdfcrop` is just `pdfcrop`. You can change that by using `pdfcrop` package option, for example:

```
\usepackage[pdfcrop=/bin/pdfcrop]{docshot}
```

`margin` When we crop the PDF rendered, we leave a margin around the content. The default value may be changed by the package option `margin`:

```
\usepackage[margin=10]{docshot}
```

`hspace` The horizontal distance between the image and its verbatim TEX source may be configured via `hspace` package option:

```
\usepackage[hspace=1em]{docshot}
```

`left` `right` The default width of the image may be changed by `left` option, while the width of the verbatim TEX source may be modified by `right` option:

```
\usepackage[left=2in,right=.5\linewidth]{docshot}
```

`dtx` If you use this package inside `.dtx` documentation, add `dtx` package option. Thanks to this option all comment symbols will be removed from line starts:

```
\usepackage[dtx]{docshot}
```

`tmpdir` The default location of temp files is `\docshots@tmpdir`. You can change this using `tmpdir` option:

```
\usepackage[tmpdir=/tmp/foo]{docshot}
```

`runs` By default, we run `pdflatex` just once for each docshot. You can change this number using the package option `runs`. This may be helpful if you need BibTEX processing, for example:

```
\usepackage[runs=3]{docshot}
```

`small` `tiny` You don't have too much freedom in formatting of verbatim texts, but you can make their font a bit smaller using `small` package option. You can also make it very small using `tiny` option:

```
\usepackage[small]{docshot}
```

`log` With `log` option you can make us print all possible logs to the main TEX log. By default, we don't do this and you won't see the output of `pdflatex` compilation, for example. Just use it like this:

```
\usepackage[log]{docshot}
```

`inputminted` By default, we render the verbatim text using `\VerbatimInput` command. You can change that and make us use `\inputminted` from minted package instead, for example:

```
\usepackage{minted}
\setminted[java]{frame=lines,framesep=2mm}
\usepackage[inputminted=java]{docshot}
```

`lstinputlisting` By default, we render the verbatim text using `\VerbatimInput` command. You can change that and make us use `\lstinputlisting` from listings package instead, for example:

```
\usepackage{listings}
\lstset{basicstyle=\small}
\usepackage[lstinputlisting]{docshot}
```

3 Prerequisites

`\docshotPrerequisite` If you need some files to be present next to the `.tex` snippet while it's rendered by `pdflatex`, you can use `\docshotPrerequisite` with a single mandatory argument. The argument is the name of a file you need to be copied from current directory to the temporary directory, where all snippets are rendered. The command can be used either in the body of the document or in the preamble — it doesn't matter, as long as it shows up before the docshot that needs the prerequisite. For example:

```
\documentclass{article}
\usepackage{docshot}
\docshotPrerequisite{duck.jpg}
\begin{document}
\begin{docshot}
  \documentclass{article}
  \usepackage{graphicx}
  \pagestyle{empty}
  \begin{document}
    This is my favorite picture of a duck:
    \includegraphics[width=2in]{duck.jpg}
  \end{document}
\end{docshot}
\end{document}
```

`\docshotAfter` If you need something to happen after each `pdflatex` run of a docshot, you may use `\docshotAfter` command right before `docshot` environment. For example, you have a bibliography file that you want to be visible for all snippets and you want all of them to run biber in order to process citations:

```
\documentclass{article}
\usepackage{docshot}
\docshotPrerequisite{main.bib}
\begin{document}
\docshotAfter{biber $2}
\begin{docshot}
  \documentclass{acmart}
  \usepackage[natbib=true]{biblatex}
  \addbibresource{main.bib}
  \pagestyle{empty}
  \begin{document}
    You must read the book of \citet{knuth1984}.
    \printbibliography
  \end{document}
\end{docshot}
\end{document}
```

The script you specify in the first argument of `\docshotAfter` will get three arguments when it runs:

`$1` the cycle of `pdflatex` processing (1, 2, ...),

`$2` the hash of the snippet,

`$3` the name of the `.tex` file.

`$2` is basically equals to `$1` with an attached `.tex` suffix. `\docshotAfter` applies only to the first `docshot` environment that goes after it! You must specify `\docshotAfter` before each `docshot` where you want such post-processing to happen.

4 Implementation

First, we include a few packages:

```
\RequirePackage{iexec}
\RequirePackage{fancyvrb}
\RequirePackage{xcolor}
\RequirePackage{graphicx}
\RequirePackage{tikz}
```

Then, we process package options:

```
\RequirePackage{pgfopts}
\def\docshots@log{}
\pgfkeys{
  /docshots/.cd,
  dtx/.store in=\docshots@dtx,
  log/.code=\def\docshots@log{log},
  lstinputlisting/.store in=\docshots@lstinputlisting,
  inputminted/.store in=\docshots@inputminted,
  tmpdir/.store in=\docshots@tmpdir,
  tmpdir/.default=_docshots,
  small/.store in=\docshots@small,
  tiny/.store in=\docshots@tiny,
  runs/.store in=\docshots@runs,
  runs/.default=1,
  pdflatex/.store in=\docshots@pdflatex,
  pdflatex/.default=pdflatex,
  gs/.store in=\docshots@gs,
  gs/.default=gs,
  pdfcrop/.store in=\docshots@pdfcrop,
  pdfcrop/.default=pdfcrop,
  margin/.store in=\docshots@margin,
  margin/.default=5,
  hspace/.store in=\docshots@hspace,
  hspace/.default=.05\linewidth,
  left/.store in=\docshots@left,
  left/.default=.3\linewidth,
  right/.store in=\docshots@right,
  right/.default=.55\linewidth,
  tmpdir,pdflatex,gs,pdfcrop,margin,hspace,left,right,runs
}
\ProcessPgfOptions{/docshots}
```

Then, we print the version of `pdflatex` to TEX log:

```
\iexec[\docshots@log,quiet]{\docshots@pdflatex\space --version}%
```

Then, we print the version of pdfcrop to TEX log:

```
\iexec[\docshots@log,quiet]{\docshots@pdfcrop\space --version}%
```

Then, we print the version of `ghostscript` to TEX log:

```
\iexec[\docshots@log,quiet]{\docshots@gs\space --version}%
```

Then, we make a directory where all temporary files will be kept:

```
\iexec[null]{mkdir -p \docshots@tmpdir/\jobname}%
```

`docshot` Then, we define `docshot` environment:

```
\newenvironment{docshot}
```

```
{\VerbatimEnvironment\begin{VerbatimOut}
  {\docshots@tmpdir/\jobname/verbatim.tex}}
{\end{VerbatimOut}%
```

If we are in `dtx` mode, leading percent characters must be removed:

```
  \ifdefined\docshots@dtx%
    \iexec[null]{perl -i -0777pe "s/(\\n|^)\\x{25} /\\1/g"
      \docshots@tmpdir/\jobname/verbatim.tex}%
  \fi%
```

We calculate MD5 hashsum of the file content:

```
  \def\hash{\pdfmdfivesum file
    {\docshots@tmpdir/\jobname/verbatim.tex}}%
```

If the PDF with the required name already exists, we ignore this step. Otherwise, we copy `verbatim.tex` into new file and run `pdflatex`:

```
  \IfFileExists{\docshots@tmpdir/\jobname/\hash.pdf}
    {\message{docshots: won't render, the PDF already exists
      (\docshots@tmpdir/\jobname/\hash.pdf)^^J}}
    {\iexec[\docshots@log,quiet]{cp \docshots@tmpdir/\jobname/verbatim.tex
      \docshots@tmpdir/\jobname/\hash.tex}%
    \foreach \n in {1,...,\docshots@runs}{
      \iexec[\docshots@log,quiet]{cd \docshots@tmpdir/\jobname;
        \docshots@pdflatex\space
        -interaction=errorstopmode
        -halt-on-error
        -shell-escape
        \hash.tex}
      \message{docshots: pdflatex run no.\n^^J}
      \IfFileExists{\docshots@tmpdir/\jobname/after.sh}
        {\iexec[\docshots@log,quiet]{chmod a+x
          \docshots@tmpdir/\jobname/after.sh}
        \iexec[\docshots@log,quiet]{cd \docshots@tmpdir/\jobname;
          ./after.sh \n\space \hash\space \hash.tex}}
        {}}}%
```

Here we delete `after.sh` which may exist here after the last compilation of `pdflatex`:

```
  \iexec[\docshots@log,quiet]{rm -f \docshots@tmpdir/\jobname/after.sh}
```

If a cropped version of the PDF with the required name already exists, we ignore this step. Otherwise, we ask `pdfcrop` to crop the PDF:

```
  \IfFileExists{\docshots@tmpdir/\jobname/\hash.crop.pdf}
    {\message{docshots: won't pdfcrop, the PDF already exists
      (\docshots@tmpdir/\jobname/\hash.crop.pdf)^^J}}
    {\iexec[\docshots@log,quiet]{\docshots@pdfcrop\space
      --margins \docshots@margin\space
      \docshots@tmpdir/\jobname/\hash.pdf
      \docshots@tmpdir/\jobname/\hash.crop.pdf}}%
```

We configure `fancyvrb`:

```
  \fvset{numbers=left,numbersep=3pt}%
  \fvset{frame=leftline,framerule=.4pt,rulecolor=\color{gray}}%
  \fvset{samepage=true}%
  \fvset{baselinestretch=1}%
  \ifdefined\docshots@small%
    \fvset{fontsize=\small}%
```

```
  \fi%
  \ifdefined\docshots@tiny%
    \fvset{fontsize=\scriptsize}%
  \fi%
```

We render the two column content:

```
  \begingroup%
  \par%
  \tikz[baseline=(a.north)]
    \node (a) [draw=gray,inner sep=0]
    {\includegraphics[width=\docshots@left]
      {\docshots@tmpdir/\jobname/\hash.crop.pdf}};%
  \hspace{\docshots@hspace}%
  \begin{minipage}[t]{\docshots@right}%
    \vspace{0pt}%
    \ifdefined\docshots@lstinputlisting%
      \lstinputlisting{\docshots@tmpdir/\jobname/\hash.tex}%
    \else\ifdefined\docshots@inputminted%
      \inputminted{\docshots@inputminted}
        {\docshots@tmpdir/\jobname/\hash.tex}%
    \else%
      \VerbatimInput{\docshots@tmpdir/\jobname/\hash.tex}%
    \fi\fi%
    \vspace{0pt}%
  \end{minipage}%
  \par%
  \endgroup%
}
```

`\docshotPrerequisite` Then, we define `\docshotPrerequisite` command:

```
\newcommand\docshotPrerequisite[1]{
  \iexec[\docshots@log,quiet]{cp #1 \docshots@tmpdir/\jobname}%
  \message{docshots: file #1 copied to
    \docshots@tmpdir/\jobname/#1^^J}%
}
```

`\docshotAfter` Finally, we define `\docshotAfter` command:

```
\newcommand\docshotAfter[1]{
  \iexec[\docshots@log,quiet]{/bin/echo -n '\detokenize{#1}'
    > \docshots@tmpdir/\jobname/after.sh}%
  \message{docshots: file
    \docshots@tmpdir/\jobname/after.sh created^^J}%
}
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.