

The everypage package*

Sergio Callegari†

2007/06/20

Abstract

The `everypage` package extends \LaTeX providing hooks to do actions on every page or on the current page. Specifically, actions are performed *before* the page is shipped, so they can be used to put watermarks *in the background* of a page, or to set the page layout. The package reminds in some sense `bobhook` by Karsten Tinnfeld, but it differs in the way in which the hooks are implemented, as detailed in the following. In some sense it may also be related to the package `everyshi` by Martin Schroeder, but again the implementation is different.

1 Introduction

This program adds two \LaTeX hooks that get run when document pages are finalized and output to the dvi or pdf file. Specifically, one hook gets executed on every page, while the other is executed for the current page. Hook actions are performed *before* the page is output on the medium, and this is important to be able to play with the page layout or to put things *behind* the page contents (e.g., watermarks such as an image, framing, the “DRAFT” word, and the like).

The package reminds in some sense `bobhook` by Karsten Tinnfeld, but it differs in the way in which the hooks are implemented:

1. there is no formatting inherent in the hooks. If one wants to put some watermark on a page, it is his own duty to put in the hook the code to place the watermark in the right position. Also note that the hooks code should *eat up no space* in the page. Again, if the hooks are meant to place some material on the page, it is the duty of the hook programmer to put code in the hooks to pretend that the material has zero width and zero height. The implementation is *lighter* than the `bobhook` one, and possibly more flexible, since one is not limited by any pre-coded formatting for the hooks. On the other hand it is possibly more difficult to use. Nonetheless, it is easy to think of other packages relying on `everypage` to deliver more user-friendly and *task specific* interfaces. Already there are a couple of them: the package `flippdf` produces mirrored pages in a PDF document and `draftwatermark` watermarks document pages.
2. similarly to `bobhook` and `watermark`, the package relies on the manipulation of the internal \LaTeX macro `\@begindvi` to do the job. However, the redefinition of `\@begindvi` is here postponed as much as possible, striving to avoid

*This file (`everypage.dtx`) has version number 1.1, last revised 2007/06/20.

†Sergio Callegari can be reached at `sergio.callegari@gmail.com`

interference with other packages using `\AtBeginDvi` or anyway manipulating `\@begindvi`. Specifically `everypage` makes no special assumption on the initial code that `\@begindvi` might contain.

Also in some sense `everypage` can be related to package `everyshi` by Martin Schroeder, but it differs radically from it in the implementation. In fact, `everypage` operates by manipulation of the `\@begindvi` macro, rather than at the lower level `\shipout` macro.

2 User interface

`\AddEverypageHook` The `\AddEverypageHook` accepts one argument and adds it to the list of hooks that are run before every page is output. The `\AddThispageHook` accepts one argument and adds it to the list of hooks that are run before the current page is output.

Note that once hooks are stacked, there is no way to unstack them, nor to clear them. In order to have hooks that get run only when specific conditions are met, conditionals must be included in the hooks.

Also note that no particular assumption is made on the \LaTeX output driver, so `everypage` should work equally well with \LaTeX and `pdf \LaTeX` . Furthermore, the package should work equally well with `dvi`, `dvips`, etc. output drivers. Obviously, the final compatibility with the different output drivers depends on the actual code that is placed in the hooks.

For usage examples, please see the `flippdf` and `draftwatermark` packages.

3 Implementation

Announce the name and version of the package, which requires \LaTeX 2 ϵ .

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{everypage}%
3 [2007/06/20 1.1 Hooks to run on every page]
```

`\sc@everypage@hook` Define internal macros to hold the hooks and initialize them to contain nothing.

```
\sc@everypage@hook 4 \newcommand{\sc@everypage@hook}{}
\sc@everypage@hook 5 \newcommand{\sc@thispage@hook}{}
\sc@everypage@hook
```

`\AddEverypageHook` Define the commands used to add the hooks.

```
\AddEverypageHook 6 \newcommand*{\AddEverypageHook}[1]{%
\AddThispageHook 7 \g@addto@macro\sc@everypage@hook{#1}}
8 \newcommand*{\AddThispageHook}[1]{%
9 \g@addto@macro\sc@thispage@hook{#1}}
```

`\sc@ep@init` The internal initialization code of the package. The package works by redefining the normal `\@outputpage` routine that takes care of outputting pages, so that the modified version first calls a special preamble, and then calls the original `\@outputpage` code and finally a postamble.

```
10 \newcommand*{\sc@ep@init}{%
11 \let\sc@op@saved\@outputpage
12 \def\@outputpage{%
13 \sc@op@preamble
14 \sc@op@saved
15 \sc@op@postamble}}
```

`\sc@op@preamble` The `outputpage` preamble contains instructions to redefine the `\@beginvdi` macro that is called at every page output by the original `\@outputpage` code. Specifically: first the original `\@beginvdi` is saved; then the hooks are called; then the hooks for the current page are cleared; eventually, the saved `\@beginvdi` is called.

```
16 \newcommand*{\sc@op@preamble}{%
17   \let\sc@beginvdi\@beginvdi
18   \def\@beginvdi{%
19     \sc@everypage@hook
20     \sc@thispage@hook
21     \gdef\sc@thispage@hook{}
22     \sc@beginvdi}}
```

`\sc@op@postamble` The `outputpage` postamble simply restores the `\@beginvdi` command to the saved value.

```
23 \newcommand*{\sc@op@postamble}{%
24   \let\@beginvdi\sc@beginvdi}
```

Note that in exceptional cases this might not be the intended behaviour. For instance, consider situations where the `\@beginvdi` is hacked by some other package to modify itself. By restoring the saved value, one might lose the modifications. This potential issue is currently under investigation.

As the very last thing, the `\AtBeginDocument` macro is called to insert the `everypage` initialization routine in the queue of commands to be executed when the actual document begins. In this way, the `everypage` initialization code is run *after* other packages are loaded.

```
25 \AtBeginDocument{\sc@ep@init}
26 \endinput
```