The afterpackage package*

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2006/01/17

The package allows specify what additional commands should be applied to a package after its loading. To describe the usability of the package, consider the task of customization of a number of packages. Let we need to apply some package-specific code after loading packages such as customization of their defaults or hooks. To do this, we need to load all required packages and then apply the related staff to them. But, sometimes loading of all required packages is impossible or unnecessary because packages can have options or some of customized packages are rarely useful.

The main idea of using this package is to specify how other packages can be customized in such a way that their customization is applied when and if customized packages are loaded. All the customization is applied in the preamble only.

`\AfterPackage` To customize a package, load the **afterpackage** package and insert any number of the following commands anywhere in the preamble:

`\AfterPackage{`⟨*package name*⟩`}{`⟨*commands*⟩`}`

The command works as follows: it tests the required package to be already loaded; if the package is loaded, it directly applies ⟨*commands*⟩; otherwise, it collects ⟨*commands*⟩ in a special macro. The specified macro will be applied later when the respective package will be loaded. This operation is executed when the package loading finishes and all end-of-package hooks are applied. So, the ⟨*commands*⟩ are applied *after the end* of respective package.

1 The Implementation

`\AfterPackage` The command tests the `\ver@`⟨*filename*⟩ command to be defined and considers either collect commands in a special macro or apply them directly.

```
1 ⟨*package⟩
2 \newcommand*\AfterPackage[1]{%
3   \expandafter\ifx\csname ver@#1.\@pkgextension\endcsname\relax
4     \@ifundefined{#1.\@pkgextension-@dd}{%
```

*This file has version number v1.1, last revised 2006/01/17.

```
      \expandafter\let\csname#1.\@pkgextension-@dd\endcsname\@empty
      \expandafter\@onlypreamble\csname#1.\@pkgextension-@dd\endcsname
    }{}%
    \wlog{After Package Info: Collect commands for #1\on@line}%
    \def\@tempa{%
      \expandafter\g@addto@macro\csname#1.\@pkgextension-@dd\endcsname}%
  \else
    \ATP@apply@info{#1}%
    \let\@tempa\@firstofone
  \fi
  \@tempa
}
\@onlypreamble\AfterPackage
```

`\@popfilename` To apply collected commands to a package, we use the `\@popfilename` hook. This hook is applied after loading a package. It pops a name of previous package or class which was active before loading a package. We save the original value of `\@popfilename` hook in a special macro and redefine it to execute the `\`⟨*filename*⟩`-@dd` hook before pop of the previous filename.

```
\let\ATP@popfilename\@popfilename
\@onlypreamble\ATP@popfilename
\def\@popfilename{%
  \@ifundefined{\@currname.\@currext-@dd}{}{%
    \ATP@apply@info{\@currname}%
    \csname\@currname.\@currext-@dd\endcsname
    \expandafter\let\csname\@currname.\@currext-@dd\endcsname\relax
  }%
  \ATP@popfilename
}
```

`\ATP@apply@info` This information message is logged when commands are applied. Its parameter contains a name of package which commands are customized.

```
\def\ATP@apply@info#1{%
  \wlog{After Package Info: Apply commands to #1\on@line}%
}
\@onlypreamble\ATP@apply@info
```

⟨/package⟩