# The **codesection** package

Matthias Pospiech
matthias@pospiech.eu

v0.1 from 2014/06/27

## 1 Introduction

This packages provides an environment to switch a section of code on or off. The code can be placed anywhere in the file and is not limited to the document or the preamble. The motivation for this package was to have commands which allow to preselect if sections of code in a preamble of a template are executed or not.

## 2 Origin of the code

The code is based on the `verbatim.sty` package and was originally modified by Ulrich Diez to match the pure comment functionality. Further modifications are contributed by Matthias Pospiech. During the development some discussion about the best approach took place on de.comp.text.tex [1], which resulted in the current code.

## 3 Usage

The idea of the following commands is to define a collection of code, here notated as a *section*, which can be executed as it would be without the commands or which is not executed at all. To use that section it must be defined with *true* (execute code) or *false* (skip code).

`\DefineCodeSection` `[`⟨*true/false*⟩`]{`⟨*name*⟩`}`

Defines a code section with a *name*. The default is *true*, thus the code will be executed.

`\SetCodeSection` `{`⟨*name*⟩`}{`⟨*true/false*⟩`}`

---

[1] `http://groups.google.com/group/de.comp.text.tex/browse_thread/thread/2c18f0c221ab167f/`

is like `\DefineCodeSection`, but with both arguments mandatory.

**`\BeginCodeSection`** `{`⟨*name*⟩`}`

starts the code section with the given name and

**`\EndCodeSection`** `{`⟨*name*⟩`}`

ends the code section with the given name. Note that both commands need to be paired and not to be nested with other code sections.

`\BeginCodeSection` and `\EndCodeSection`

mimic an environment. It would be preferable to define them as an environment, but that opens a group in TEX, which has many disadvantages. For example this would make it impossible to load packages. Therefore this package defines paired commands and consequently, has no such limitations.

## 4 Example

In the following code the first section is going to be executed and the second and the third are completely skipped.

```
\DefineCodeSection[true]{ExecuteMe}
\DefineCodeSection[false]{SkipMe}
%
\BeginCodeSection{ExecuteMe}
  This sentence has ...
\EndCodeSection{ExecuteMe}
%
\BeginCodeSection{SkipMe}
  no end.
\EndCodeSection{SkipMe}
%
\SetCodeSection{ExecuteMe}{false}
%
\BeginCodeSection{ExecuteMe}
  a different ending.
\EndCodeSection{ExecuteMe}
```

This sentence has ...

## 5 Implementation

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
\ProvidesPackage{codesection}
          [2014/06/27  v0.1  disableable code sections]
```

```
\RequirePackage{etoolbox}
%
```

\DefineCodeSection

```
%%------------------------------------------------------------------------
%% provide new if (\ifCodeSection<name>)
%% with definition \CodeSection<name><true>
%%
\newcommand{\DefineCodeSection}[2][true]{%
  \expandafter\newif\csname ifCodeSection#2\endcsname
  \csname CodeSection#2#1\endcsname
}%
%
```

\SetCodeSection

```
%%------------------------------------------------------------------------
%% Alternative to \DefineCodeSection
\newcommand{\SetCodeSection}[2]{%
  \DefineCodeSection[#2]{#1}
}%
%
```

\BeginCodeSection

```
%%------------------------------------------------------------------------
\newcommand\BeginCodeSection[1]{%
  \ifcsdef{ifCodeSection#1}{}{%
    \PackageError{codesection}{Section #1 is unknown\MessageBreak}{}%
  }
  \csname ifCodeSection#1\endcsname
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi
  {% comment all code inside template section
    \@bsphack
%% open new group
    \begingroup
%% save current template section name
    \def\@currtemplate{#1}%
    \let\do\@makeother\dospecials
    \catcode`\^^M\active
%% enter main loop
    \codeSection@
  }%
  {% execute all code inside template section
%% = do nothing except trimming spaces
    \@bsphack\@esphack%
  }%
}
%
```

\EndCodeSection

```
%%--------------------------------------------------------------------
%%  The macros \@bsphack and  \@esphack are internal to LaTeX;
%%  they ensure that an entity like a marginal note or
%%  label definition does not introduce any extra space
%%  into a paragraph, independently of whether or not
%%  it is attached to a word.
\newcommand\EndCodeSection[1]{\@bsphack\@esphack}
%
```

Modified code from `verbatim.sty`. This code is not very well documented, because I do not understand it well enough.

```
%%--------------------------------------------------------------------
%% usage ???
\@ifundefined{vrb@catcodes}{%
  \def\vrb@catcodes{%
     \catcode`\!12\catcode`\[12\catcode`\]12%
  }%
}{}%
%%--------------------------------------------------------------------
\begingroup
\vrb@catcodes
\lccode`\!=`\\
\lccode`\[=`\{
\lccode`\]=`\}
\catcode`\~=\active
\lccode`\~=`\^^M
\lccode`\C=`\C
\lowercase{%
%%--------------------------------------------------------------------
  \def\codeSection@#1{%
    \endgroup
%% ----
    \def\codeSection@##1~{\codeSection@@##1!#1\@nil}%
%% ----
    \def\codeSection@@##1!#1{\futurelet\next\codeSection@@@}%
%% ----
    \def\codeSection@@@##1\@nil{%
       \ifx\next\@nil
         \let\next\codeSection@
       \else
         \def\@tempa####1!#1\@nil{####1}%
         \def\next{\expandafter\codeSection@test\@tempa##1\@nil~}%
       \fi
       \next
    }%
%% ----
    \def\codeSection@test##1{%
      \let\next\codeSection@test
      \if\noexpand##1\noexpand~\let\next\codeSection@
      \else \if\noexpand##1
      \else \if\noexpand##1\noexpand[\let\@tempc\@empty
                                     \let\next\codeSection@testend
      \else \def\next{\codeSection@##1}%
      \fi\fi\fi
      \next
    }%
%% ----
    \def\codeSection@testend##1{%
      \if\noexpand##1\noexpand~\let\next\codeSection@
      \else \if\noexpand##1\noexpand]\let\next\codeSection@@testend
      \else\if\noexpand##1\noexpand!\def\next{\codeSection@!}%
      \else \expandafter\def\expandafter\@tempc\expandafter
            {\@tempc##1}%
      \fi\fi\fi
      \next
    }%
%% ---- test if end statement belongs to current section
%%      saved in \@currtemplate
    \def\codeSection@@testend{%
      \ifx\@tempc\@currtemplate
%% end group and call rescan
         \edef\next{\noexpand\endgroup\noexpand\@esphack
                    \noexpand\codeSection@rescan{\@currtemplate}}%
       \else
%% start loop gain
         \let\next\codeSection@
       \fi
       \next
    }%
%% ---- does what ???
    \def\codeSection@rescan##1##2~{%
      \if\noexpand~\noexpand##2~%
      \else
        \@warning{%
          Characters dropped after
          `\string\EndCodeSection{##1}'%
        }%
      \fi
    }%
  }%
%%----------------------------------------------------------------------
} % lowercase
\codeSection@{EndCodeSection}%
%%----------------------------------------------------------------------
%
```

# 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.