The *standalone* Class and Package

Martin Scharrer

martin@scharrer-online.de

http://www.ctan.org/pkg/standalone/

Version v0.1 – 2010/03/21

1 Introduction

Larger LaTeX documents can be split into multiple TeX files which are then included in a main document with `\include` for e.g. chapter files or `\input` for e.g. TeX-coded pictures. Keeping pictures in their own sub-files improves readability of the main file and simplifies the sharing of them between different documents. However, during the, sometimes lengthly, drawing/coding process it has benefits to be able to compile the pictures on their own. The compile process is much quicker and the resulting document only holds the picture which avoids constant page turning and zooming.

While it is possible to write a small 'main' file for each picture file, this method is a little cumbersome and clutters the directories with a lot of extra files. A second method is to place the 'main' components, i.e. a preamble, directly into the picture files and make the main document ignore this code sections.

The package `standalone` can be used in the main document to skip all extra preambles in included files. A `standalone` class is also provided to minimise the extra preamble code needed in this files. It's usage is optional, but simplifies and standardises how picture files are compiled standalone. The class uses by default the `preview` package to create a output file which only contains the picture with no extra margins, page numbers or anything else. Both the class and the package read a configuration file `standalone.cfg` which allows the user to adjust settings and macros easily on a per directory base.

Similar Packages

The `docmute` package by T.M. Trzeciak is written with the same function as the `standalone` package. However, no special class file or configuration file is provided.

2 Usage

2.1 Quick instructions

Load the `standalone` *package* and all packages needed by all the sub-files in the main document and include your picture or other sub-files using `\input` as normal.

In the sub-files use the `standalone` *class* using the normal `\documentclass` and load all packages needed for the particular file. Finally wrap the actual file content in a `document` environment.

When the sub-file is compiled on its own the `\documentclass` and `document` environment will be active as normal. The main file, however, will skip everything from the `\documentclass` till the `\begin{document}`. The (now fake) `document` environment is redefined to be a simple TeX-group. Any code after the `\end{document}` will be ignored. The real `document` environment of the main file will be unaffected and will work as normal.

2.2 Class Options

The `standalone` class will load a real document class. By default this is `article`. The document class normally has not much influence on sub-files like pictures, especially when the `preview` package is active. However, the used class can be adjusted by the user with the `class=`⟨*class name*⟩ option.

A special `beamer` option is provided to handle beamer frames and overlays correctly. See section 2.6 for more information.

All other used options are passed to the loaded class.

2.3 Package Options

At the moment the `standalone` package does not provide any options.

2.4 Environments and Macros

`standalone` The `standalone` environment is wrapped around the content of each sub-file when compiled standalone. By default it only contains a `preview` environment as long the class is not called with the `preview=false` option. It can be redefined in the configuration file if required. When compiled as part of a main document the `standalone` environment does nothing (apart of being a TEX group).

`standaloneframe` The `standaloneframe` environment is only defined when the class is called with the `beamer` option and acts as a replacement of the `frame` environment of beamer when compiled standalone. All optional arguments of `frame` are supported. When compiled as part of a main document it does nothing except of gobbling its arguments.

`\ifstandalone` Both the class and the package provide the if-switch `\ifstandalone`, which can be used to only include code if the file is compiled standalone. The switch is set to `\iftrue` by the class and to `\iffalse` by the package.

The additional file `standalone.tex` also defines this switch by checking if `\documentclass` was already used. It can be included with `\input{standalone}` and is intended for specialised files which do not use the `standalone` class.

Example 1: Use of *standalone* class.

```
% A sub-file (e.g. picture) using the 'standalone' class:
```

```
% Use 'standalone' as document class:
\documentclass{standalone}

% Load packages needed for this TeX file:
\usepackage{tikz}

% Surround TeX code with 'document' environment as usually:
\begin{document}
% Add your TeX code, e.g. a picture:
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\end{document}
```

Example 2: Effective code if compiled standalone.

```
\documentclass{article}

\newenvironment{standalone}{\begin{preview}}{\end{preview}}
\input{standalone.cfg}
% which by defaults loads: \PassOptionsToPackage{active,tightpage}{preview}
\usepackage{preview}

\usepackage{tikz}

\begin{document}
\begin{standalone}
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\end{standalone}
\end{document}
```

Example 3: Effective code if included in a main document.

```
\begingroup
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1) node [midway] {Example};
\end{tikzpicture}
\endgroup
\endinput
```

2.5 Usage of the package

Example 4: Use of *standalone* package.

```
% Main file
% Real document class:
\documentclass{article}

% Use the 'standalone' package:
\usepackage{standalone}

% Load all packages needed for all sub-files:
\usepackage{tikz}

% Inside the real 'document' environment read the sub-file with '\input'
\begin{document}
% ...
\begin{figure}
  \input{subfile}
  \caption{A subfile}
\end{figure}
% ...
\end{document}
```

2.6 Support for Beamer Presentations

Presentation can be written in LaTeX using the `beamer` class. Each presentation frame is wrapped in a `frame` environment. Overlay effects can be added using special macros. This effects result in multiple pages per frame. Pictures with such overlay effects can not be compiled standalone using the normal settings. Instead the `standalone` class must load the `beamer` class and wrap the content also in a `frame` environment while skipping the `preview` environment. To activate this settings load the `standalone` class with the `beamer` option. Because the `frame` environment is quite special (it normally collects all it's content and calls the `\frame`) and must also support verbatim content it is not easily possible to redefined the `document` environment to include `frame`. Also `frame` accepts options which `document` doesn't. Therefore a second environment called `standaloneframe` is used in the beamer picture files. It will be equal to `frame` in standalone mode, but without effect otherwise.

Example 5: Use of *standalone* class.

```
% Use of 'standalone' class with a beamer overlay:
\documentclass[beamer]{standalone}

% Load packages needed for this TeX file:
\usepackage{tikz}

% Surround TeX code with 'document' environment as usually:
```

```
\begin{document}
\begin{standaloneframe}[options, e.g. 'fragile' for verbatim content]
 % Add your TeX code:
   \only<1>{ One }%
   \only<2>{ Two }%
\end{standaloneframe}
\end{document}
```

Example 6: Effective beamer code if compiled standalone.

```
\documentclass{beamer}

\input{standalone.cfg}

\usepackage{tikz}

\begin{document}
\begin{frame}[your options]
   \only<1>{ One }%
   \only<2>{ Two }%
\end{frame}
\end{document}
```

Example 7: Effective code if included in a beamer presentation.

```
\begingroup
   \only<1>{ One }%
   \only<2>{ Two }%
\endgroup
\endinput
```

2.7 standalone.tex

Example 8: Usage of 'standalone.tex'.

```
\input{standalone} % use before any '\documentclass'
\ifstandalone
  % Used only if compiled standalone
\fi
```

3 Implementation

3.1 The Package File

```
\expandafter\newif\csname ifstandalone\endcsname
\standalonefalse
```

\sa@documentclass

```
\newcommand{\sa@documentclass}[2][]{%
  \let\document\sa@document
  \expandafter\sa@@documentclass
}
```

\sa@@documentclass

```
\def\sa@@documentclass{%
  \begingroup\def\sa@gobbleto{document}\sa@gobble
}
```

\sa@gobble

```
\long\def\sa@gobble#1\begin#2{%
  \def\@tempa{#2}%
  \ifx\@tempa\sa@gobbleto
    \def\next{\expandafter\endgroup\expandafter\begin\expandafter{\sa@gobbleto}}%
  \else
    \def\next{\sa@gobble}%
  \fi
  \next
}

\@ifundefined{standalone}
  {\newenvironment{standalone}[1][]{}{}}
  {}
```

Gobbles all arguments: `<...>[<...>][...]{...}{...}`. Please note that the last two `{ }` arguments are also optional.

```
\@ifundefined{standaloneframe}
  {\@ifundefined{beamer@newenv}
    {\newenvironment{standaloneframe}[1][]{%
      \@ifnextchar[%]
        {\sa@framegobbleopt}{\sa@framegobbleargs}}{}%
    }
    {\newenvironment<>{standaloneframe}[1][]{%
      \@ifnextchar[%]
        {\sa@framegobbleopt}{\sa@framegobbleargs}}{}%
    }
   \def\sa@framegobbleopt[#1]{\sa@framegobbleargs}
   \def\sa@framegobbleargs{%
     \@ifnextchar\bgroup
       {\sa@framegobbleargs@}%
       {}%
```

```
    }
    \def\sa@framegobbleargs@#1{%
     \@ifnextchar\bgroup
       {\@gobble}%
       {}%
    }
   }
   {}
```

`\sa@orig@document`

```
\let\sa@orig@document\document
% \end{macro}
%
% \begin{macro}{\sa@orig@enddocument}
%    \begin{macrocode}
\let\sa@orig@enddocument\enddocument
```

`\document`

```
\def\document{%
  \sa@orig@document
  \let\documentclass\sa@documentclass
  \ignorespaces
}
```

`\sa@document`

```
\def\sa@document{%
  \let\enddocument\sa@enddocument
  \sa@atbegindocument
}
```

`\sa@enddocument`

```
\def\sa@enddocument{%
  \sa@atenddocument
  \aftergroup\sa@@enddocument
}
```

`\sa@atbegindocument`

```
\def\sa@atbegindocument{%
  \ignorespaces
}%
```

`\sa@atenddocument`

```
\def\sa@atenddocument{%
  \unskip
}%
```

`\sa@@enddocument`

```
\def\sa@@enddocument{%
  %\let\document\sa@orig@document
```

```
  \let\enddocument\sa@orig@enddocument
  \endinput
}
```

\sa@processpreamble

```
\def\sa@processpreamble{%
  \renewcommand\usepackage[2][]{%
    \message{^^J%
      INFO: Sub-file requires the following package(s):^^J
      \space\space[##1]{##2}^^J%
    }%
  }%
  \let\RequirePackage\usepackage
}

%%\def\sa@@documentclass{\sa@processpreamble}
```

3.2 The Class File

```
\def\sa@classoptionslist{}
\RequirePackage{kvoptions}
\SetupKeyvalOptions{prefix=sa@}
\DeclareBoolOption[true]{preview}
```

standalone

The `standalone` environment is defined by default to be without effect. The `\endstandalone` macro is set to `\relax`, so a redefinition with `\renewenvironment` can be detected later.

```
\let\standalone\empty
\let\endstandalone\relax
```

\sa@cls@document
\sa@cls@enddocument

```
\def\sa@cls@document{\standalone}
\def\sa@cls@enddocument{\endstandalone}
```

The `beamer` option defines the `standalone` environment as a replacement of `frame`.

```
\DeclareVoidOption{beamer}{%
  \def\sa@class{beamer}%
  \sa@previewfalse
  \newenvironment{standaloneframe}{%
    \@ifnextchar<%
      {\@standaloneframe}%
      {\@@standaloneframe{}}%
  }{\end{frame}}%
  \def\@standaloneframe<##1>{%
      \@@standaloneframe{<##1>}%
  }
  \def\@@standaloneframe##1{%
    \@ifnextchar[%]
```

```
      {\@@@standaloneframe{##1}}%
      {\@@@standaloneframe{##1}[]}%
  }%
  \def\@@@standaloneframe##1[{%
    \@ifnextchar<%
      {\@@@@standaloneframe{##1}[]%
      {\@@@@@@standaloneframe{##1}[]%
  }%
  \def\@@@@standaloneframe##1[##2]{%
    \@ifnextchar[%]
      {\@@@@@standaloneframe{##1}{##2}}%
      {\begin{frame}##1[##2][environment=standaloneframe]}%
  }%
  \def\@@@@@standaloneframe##1##2[##3]{%
      \begin{frame}##1[##2][environment=standaloneframe,##3]%
  }%
  \def\@@@@@@standaloneframe##1[##2]{%
      \begin{frame}##1[environment=standaloneframe,##2]%
  }%
}
\DeclareStringOption[article]{class}
\DeclareStringOption[]{frameoptions}
\DeclareDefaultOption{%
  \xdef\sa@classoptionslist{\sa@classoptionslist,\CurrentOption}%
}
\input{standalone.cfg}
\ProcessKeyvalOptions*\relax
\let\@classoptionslist\sa@classoptionslist
\xdef\@tempa{[\sa@classoptionslist]{\sa@class}}
\expandafter\LoadClass\@tempa
\@nameuse{sa@afterclassloaded}
```

Preview Code

The `standalone` environment is redefined to use the `preview` environment as long it was not redefined in the configuration file.

```
\ifsa@preview
  \@ifundefined{endstandalone}{%
    \renewenvironment{standalone}
      {\preview   }
      {\endpreview}
  }{}%
  \RequirePackage{preview}
\fi
\RequirePackage{standalone}[2010/03/21]
\standalonetrue
```

`\document`

```
147 \def\document{%
148   \sa@orig@document
149   \let\documentclass\sa@documentclass
150   \sa@cls@document
151 }
```

\enddocument

```
152 \def\enddocument{%
153   \sa@cls@enddocument
154   \sa@orig@enddocument
155 }
```

3.3 Simple TeX File

```
156 \expandafter\ifx\csname ifstandalone\endcsname\relax
157 \expandafter\newif\csname ifstandalone\endcsname
158 \expandafter\ifx\csname @twoclasseserror\endcsname\documentclass
159 \else
160   \standalonetrue
161 \fi
162 \fi
```

3.4 Config File

Default content of the configuration file. Users can place there own `standalone.cfg` in their `texmf` directory or in the local document directory to define their own settings as described above. To load the default config file from a user config file use `\input{standalone/standalone.cfg}`.

```
163 \PassOptionsToPackage{active,tightpage}{preview}%
```