tsvtemplate

a1	b1	c1
a2	b2	c2
a3	b3	c3

author	Esger Renkema
contact	minim@elrenkema.nl

This is a small package with a single purpose: reading tab-separated value (tsv) files and applying a template to the content of their rows.

Loading this package

Plain tex users can say `\input tsvtemplate`; latex users can say `\usepackage {tsvtemplate}`. You must use the lua(la)tex compiler.

Using this package

The main command of this package is `\tsvtemplate`, used like this

```
\tsvtemplate [names,of,the,columns]
        {... template ...} {filename.tsv}
```

This will read the file named `filename.tsv` row-by-row and assign the contents of the fields of every row to the control sequences `\names \of \the \columns`. These control sequences can then be used in the `template`, which will be repeated for every row.

If you do not specify column names, they will be read from the first line of the file. In that case spaces will be ignored, so that a column headed by e.g. `Postal Code` is available as `\PostalCode`.

In the latex case, an environment is also available as

```
\begin{tsv template}[names,of,the,columns]{filename.tsv}
    ... template ...
\end{tsv template}
```

Field separators

You can use other field separators than tabs by using the lower-level command `\applytemplate`; in fact, `\tsvtemplate[fields]` is equivalent to

```
\applytemplate \^^I {fields}
```

(Note that `\^^I` is just the tab character, preceded by a backslash. For colon-separated values you could say `\:` etc.) The `fields` may be separated either by commas or by the separator character. Note that they are no longer optional: leave the argument empty in order to have tsv read the first line of the file as a header line.

Limitations

Every row in the tsv file must have the exact same number of columns; an error is emitted otherwise.

When reading files with commas or semicolons as field separators, be aware of programs generating fields surrounded by double quotes (`"` characters) containing commas (or semicolons) themselves. Such files will confuse this package. The best thing you can do is convert them to proper tsv, which has no need for quoting.

Implementation

The latex style file simply loads the plain tex implementation:

—— file: tsvtemplate.sty

```
% An environment for \LaTeX.
\ProvidesPackage{tsvtemplate} \RequirePackage{environ} \input tsvtemplate
\NewEnviron{tsv template}[2][]{\applytemplate\^^I{#1}{\BODY}{#2}}
```

. . . which has been kept quite short, so that you may the more easier copy and
modify it to suit your own needs:

—— file: tsvtemplate.tex

```
% Do not read this file twice.
\ifdefined\tsvloaded\expandafter\endinput\fi
\chardef\tsvloaded\catcode`\& \catcode`\&=11

% Implement a tsv reader.
\long\def\applytemplate#1#2#3#4{\begingroup\openin\&r#4
  \def\&w##1##2{\errmessage{#4: ##1 fields at line \the\&o\space
  (##2 discarded)}}\catcode`#112 \errorcontextlines0 \ifx\&z#2\&z
    {\catcode`\ 9 \global\read\&r to\&f}\else {\catcode`\,13
    \&e\edef\scantextokens{,}{\Uchar`#1}\global\&d{&f}{#2}}\fi
    \&e\&b\Uchar`#1{#3}}
\long\def\&d#1#2{\&e\edef\csname#1\endcsname{\scantextokens{#2}}}
\let\&e\expandafter\newcount\&o\newcount\&p\def\&q{\par}\newread\&r\def\&z{\&z}
\def\&s#1{\&w{Not enough}{line}} \def\&t#1\&z#2{\&w{Too many}{fields}}
\def\&a#1{\advance#11 } \long\def\&b#1#2{\&o0 \&p0
  \def\&x##1#1{\ifx\&z##1\&z\let\&x\relax\else\&a\&p\&d{&\the\&p}{##1}\fi\&x}%
  \&e\&x\&f#1#1\chardef\&m\&p \long\def\&x##1#1{\&a\&p \ifx\&z##1\&e\&s\else
    \&d{\csname &\the\&p\endcsname}{##1}\ifnum\&p=\&m\&e\&y\fi\fi\&x}%
  \long\def\&y\fi\&x##1#1{\fi{#2}\ifx\&z##1\relax\else\&e\&t\fi}\&p0
  \def\&n{\read\&r to\&l \ifx\&q\&l\closein\&r\let\&n\relax
    \else\&a\&o{\&e\&x\&l#1\&z#1}\fi\&n}\&n\endgroup}

% A shorthand for Plain \TeX.
\def\tsvtemplate{\futurelet\&px\&pa}
\def\&pc{\&pb[]}\def\&pb[#1]{\applytemplate\^^I{#1}}
\def\&pa{\ifx\&px[\&e\&pb\else\&e\&pc\fi}
\catcode`\&=\tsvloaded
```

Licence