The usebib package*

Enrico Gregorio
Enrico dot Gregorio at univr dot it

April 13, 2012

1 Introduction

With biblatex it's possible to use any element of a bibliographic item in a document. With packages such as natbib or simply with the native BibTeX support one can't refer directly to the title of a cited article or book without directly typing it in.

This package allows this by reading in a special environment a `.bib` file. It should be considered no more than a devious usage of a `.bib` file, to be honest. But it works.

2 Usage

The package *must* be loaded after hyperref, but it doesn't require it. It depends only on url and keyval, which should be present on every TeX distribution.

2.1 Preamble commands

`\bibinput` The main command is `\bibinput`, which accepts the name of a `.bib` file (*without* the extension) and may be repeated any number of times to read all the necessary files.[1] The command should appear only in the preamble.

`\newbibfield` With the macro `\newbibfield`, to be issued *before* `\bibinput`, it's possible to declare new known fields, for example

```
\newbibfield{annotation}
```

and say in the document `\usebibentry{xyz}{annotation}`. It's also possible to say `\newbibfield{author}`, but as author names are input in a very peculiar way, the result would be at least questionable in the vast majority of cases. This command should appear before `\bibinput`.

`\newbibignore` The entry types `@preamble` and `@string` are ignored by default. If some other type should also, it's possible to say

*This document corresponds to newunicodechar v1.0a, dated 2012/04/13.

[1] Making it dependent on the `\bibliography` command would be too difficult.

`\newbibignore{`⟨*type*⟩`}`

so that, for example, `\newbibignore{misc}` will ignore all `@misc` entry types. This command should appear before `\bibinput`.

2.2 Document commands

`\usebibentry` The main document command is `\usebibentry` which requires two arguments, the bibliography key and the field name:

`\usebibentry{`⟨*key*⟩`}{`⟨*field*⟩`}`

The ⟨*key*⟩ is what's used in the argument to `\cite` (or variations thereof) and ⟨*field*⟩ can be `title`, `year`, `note` or `isbn`, but it's possible to augment the list of known fields. So, in order to get in the document the referenced item's title one can say

```
In the book ``\usebibentry{newton1687}{title}''~\cite{newton1687},
published in~\usebibentry{newton1687}{year}, Newton
describes the Law of Universal Gravitation
```

to get the text

> In the book "Philosophiae Naturalis Principia Mathematica" [3], published in 1687, Newton describes the Law of Universal Gravitation

assuming that in the `.bib` file that has been read with `\bibinput` there's an entry such as

```
@book{newton1687,
 author={Newton, Isaac},
 title={Philosophiae Naturalis Principia Mathematica},
 publisher={Royal Society},
 address={London},
 year={1687},
 url={http://royalsociety.org},
}
```

Unknown keys and missing values trigger errors: so `\usebibentry{newton1687}{isbn}` would result in the error

```
Undefined key 'newton1687' or empty value for 'isbn'
```

Unsupported fields will be ignored and print nothing.

`\usebibentryurl` If the `.bib` entry has a `url` field, its value can be obtained with

`\usebibentryurl{`⟨*key*⟩`}`

When hyperref is not loaded, this is equivalent to calling `\url|`⟨*url*⟩`|`. In case the URL contains the character `|`, this call will produce wrong results and one can specify a different delimiter

`\usebibentry[!]{`⟨*key*⟩`}`

where, of course, the optional argument is a character not appearing in the URL. This optional argument is not needed when hyperref is loaded and, if present, it's ignored.

3 Limitations

The field delimiters in the `.bib` files *must* be braces, as must also the entry delimiters. BibTeX allows double quotes in for delimiting fields and parentheses for the entry. So, while

```
@book(newton1687,
 author="Newton, Isaac",
 title="Philosophiae Naturalis Principia Mathematica",
 publisher="Royal Society",
 address="London",
 year=1687,
 url="http://royalsociety.org",
)
```

would be accepted by BibTeX, it won't be by usebib. Supporting this is out of question, since we parse entries as a key-value list. Strictly speaking, double quotes won't give errors, as long as no comma is in the fields, but will be part of the replacement text anyways. Numeric field values can, of course, be input without field delimiters.

Another limitation is in the fact that field names should always be lowercase; supporting mixed case for these is beyond the limits of a small package like this. Sorry.

Note also that all the bibliographic data are loaded in memory, so large `.bib` files could exceed TeX capacity.

Don't ever try typing `\usebibentry{foo}{url}`: you've been warned. There's `\usebibentryurl{foo}` for this.

Finally, the package requires ε-TeX, which shouldn't be a problem, nowadays.

4 Implementation

After the usual presentation, we check first of all that the typesetting engine is sufficiently recent to include ε-TeX extensions.

```
\@ifundefined{eTeXversion}
  {\PackageError{usebib}{LaTeX engine too old, aborting}
   {Please upgrade your TeX system}\@@end}{}
```

We start by loading the required packages.

```
\RequirePackage{url}
\RequirePackage{keyval}
```

When we read a `.bib` file the `@` character is made active. Its action is to read what follows up to the next open brace, using it as the value of the `currenttype` key. This is necessary for ignoring the entry types `@preamble` and `@string` which would mess up everything.

Now we define the known fields as keys for keyval and the two auxiliary keys. The `\define@reuse@key` macro is just syntactic sugar.

```
\define@key{usebib}{currenttype}{\def\@tempa{#1}}
```

```
\define@key{usebib}{currententry}{\def\reuse@current{#1}}
\def\define@reuse@key#1{%
  \define@key{usebib}{#1}{\global\@namedef{reuse@\reuse@current @#1}{##1}}}
\define@reuse@key{title}
\define@reuse@key{isbn}
\define@reuse@key{url}
\define@reuse@key{year}
\define@reuse@key{note}
```

`\reuse@find` The first macro to be used when looking at an entry, later we'll define the active `@` to be this macro: it takes as argument the entry's type and does, for instance, `\setkeys{usebib}{currenttype=BOOK}`, but inside a `\lowercase` so that the type will be normalized. Then it checks whether `\reuse@type@book` is defined, which it isn't, so `\reuse@extract` will be expanded; since we later define `\reuse@type@preamble` and `\reuse@type@string`, such entries will be ignored because of `\@gobble`. The expansion of `\reuse@extract` is done in a group where `@` has category code 12 and also the percent character, because it can be in some URL (and has not a special meaning in BibTeX entries).

```
\def\reuse@find#1#{%
  \lowercase{\setkeys{usebib}{currenttype=#1}}%
  \ifcsname reuse@type@\@tempa\endcsname
    \expandafter\@gobble
  \else
    \begingroup\@makeother\%\makeatother
    \expandafter\reuse@extract
  \fi}
```

`\reuse@extract` The entire entry is the argument to `\reuse@extract`; since it has a fixed form, we feed the citation key as the value of `currententry` and then read all the fields as key-value pairs. The effect of `currententry=xyz` is to define `\reuse@current` to `xyz`; so a subsequent `year=2012` will do the equivalent of

```
\gdef\reuse@xyz@year{2012}
```

and similarly for the other known fields. The group opened by `\reuse@find` is then closed.

```
\def\reuse@extract#1{\setkeys{usebib}{currententry=#1}\endgroup}
```

`\reuse@type@preamble` `\reuse@type@preamble` `\newbibignore` Here we define the two entry types to ignore and the user level command to augment the list of ignored fields.

```
\let\reuse@type@preamble\@empty
\let\reuse@type@string\@empty
\def\newbibignore#1{\expandafter\let\csname reuse@type@#1\endcsname\@empty}
```

`\reuse@error` A macro for issuing an error message.

```
\def\reuse@error#1#2{%
  \PackageError{usebib}
    {Undefined key `#1' or empty value for `#2'}
    {The key you used is wrong or the value to `#2' has not been set}}
```

Next we define the other four user level macros.

\newbibfield `\newbibfield{type}` defines a new key.

```
\newcommand\newbibfield[1]{\define@reuse@key{#1}}
```

\usebibentry
\usebibentryurl
\reuse@usebibentryurl

Both `\usebibentry` and `\usebibentryurl` check whether the corresponding inner macro is defined. The latter then expands a special inner version that's defined in different ways depending whether hyperref has been loaded or not. If it is, a simple `\scantokens` hands the job to the `\url` macro, which is cleverly defined for accepting special characters; otherwise we need to be sure that the URL is properly delimited.

```
\def\usebibentry#1#2{\@ifundefined{reuse@#1@#2}
  {\reuse@error{#1}{#2}}
  {\@nameuse{reuse@#1@#2}}}
\newcommand{\usebibentryurl}[2][|]{\@ifundefined{reuse@#2@url}
  {\reuse@error{#2}{url}}
  {\reuse@usebibentryurl{#1}{#2}}}
\@ifpackageloaded{hyperref}{\@tempswatrue}{\@tempswafalse}
\if@tempswa
  \def\reuse@usebibentryurl#1#2{%
    \scantokens{\url{\csname reuse@#2@url\endcsname}\endinput}}
\else
  \def\reuse@usebibentryurl#1#2{%
    \toks@=\expandafter\expandafter\expandafter
      {\csname reuse@#2@url\endcsname}%
    \scantokens\expandafter{%
      \expandafter\url\expandafter#1\the\toks@#1\endinput}}
\fi
```

\bibinput

The macro that starts the reading of a `.bib` file. We start a group where end-of-line characters are treated like spaces and where the error messages of keyval and xkeyval about undefined keys are disabled. Then we define the active `@` to be `\reuse@find` and input the `.bib` file, closing the group at the end.

```
\newcommand{\bibinput}[1]{%
  \begingroup
  \catcode`\^^M=10
  \let\KV@err=\@gobble
  \let\KV@errx=\@gobble
  \let\XKV@err=\@gobble
  \begingroup\lccode`\~=`\@
    \lowercase{\endgroup\let~}\reuse@find
  \catcode`\@=\active \input{#1.bib}\endgroup}
```

Change History

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.