The `luabibentry` package

Oliver Kopp*

Version 0.1a as of 2012/02/02

Typically, bibliographic entries are put at the end of a document. This package allows for *repeating* bibliographic entries in the document itself. The package is inspired by bibentry, which provides similar functionality for LaTeX.

Contents

1 Introduction

This package allows one to place bibliographic entries anywhere in the text. It is to be used to produce annotated bibliographies, such as

> For an intoduction to the topic of workflow management, see Frank Leymann and Dieter Roller. *Production Workflow – Concepts and Techniques.* Prentice Hall PTR, 2000.

The idea is that the full reference is used, not just the citation [1] or Leymann and Roller [2000].

This package is a variant of bibentry.sty by Patrick W. Daly. bibentry.sty is distributed with the natbib package[1]. This documentation of luabibentry is mostly adapted from Patrick's documentation of bibentry. bibentry itself is part of the natbib package.

---

*oliver.kopp@googlemail.com

[1] `http://mirror.ctan.org/macros/latex/contrib/natbib/`

The main reason for the reimplementation is the incompatibility of hyperref's backref with bibenetry.sty. The `\saved @bibitem` solution did not work here.

The differences to bibentry.sty is: The commands `\nobibliography` and `\nobibliography*` are unsupported. luabibentry always uses the bibliography of the document.

2 Usage

`\setupbibentry{<bibliography>}` before the usage of `\bibentry`. `\setupbibentry{\jobname}` may be used if the bibliography has the same name as the `.tex` file.

`\bibentry{<entry>}` where you want to have placed an entry. In case an entry is not found, "?" is output.

3 Caveats

The caveats of the entry format are similar to the bibentry package. The only difference is that luabibentry expects the key as last token in the bibitem entry. Thus, the following text is a verbatim copy of bibentry's documentation with the reference to the allowed space after the key being removed.

The entries in the `.bbl` must be of the form

> `\bibitem[`⟨*label*⟩`]{`⟨*key*⟩`}`
> *Text of the reference entry.*
>
> `\bibitem`...

That is, there must be a new line after the `{`⟨*key*⟩`}` and a blank line before the next `\bibitem`. The final period in the text will be removed, if present, allowing one to place the `\bibentry` commands in mid-sentence. Of course, there may be other periods within the text that might look funny.

4 Test

A simple test whether luabibentry runs is provided here:

```
\documentclass{article}
\usepackage{luabibentry}
\setupbibentry{\jobname}

\usepackage[backref=page]{hyperref}

\begin{document}

The entry for \cite{LR2000} is: \bibentry{LR2000}.

```

```
\bibliographystyle{plain}
\bibliography{test-luabibentry}

\end{document}
```

```
@BOOK{LR2000,
  title = {{P}roduction {W}orkflow -- {C}oncepts and {T}echniques},
  publisher = {Prentice Hall PTR},
  year = {2000},
  author = {Frank Leymann and Dieter Roller},
  isbn = {0130217530}
}
```

5 Implementation of Lua Module `luabibentry.lua`

```
module("luabibentry", package.seeall)
require("lualibs-file")

-- stores all entries
local entries = {}

-- builds the data by reading the given filename
function builddata(filename)
  -- Parameters seem to be passed as arrays.
  -- We access the first element of the parameter to get the filename
  local file = io.open(filename[1], "r")
  if file==nil then
     texio.write_nl("luabibentry: could not open file " .. filename[1])
     return
  end
  local line = file:read("*line")
  while (line~=nil) do
    -- \bibitem is our marker for new entries
    local i = string.find(line, "\\bibitem")
    if i~=nil then
      -- we expect the key in brackets in the same line
      i = string.find(line,"{")
      local lasti = 0
      -- we jump to the last bracket
      while i~= nil do
         lasti = i
         i = string.find(line,"{",i+1)
      end
      local key = string.sub(line, lasti+1)
      -- we use the text from the last opening bracket ("{") until
      -- the end of the line minus one
      -- we expect nothing more to follow in this line
      key = string.sub(key, 1, string.len(key)-1)
      -- the next lines are the entry
      -- we expect an entry to be finished with a blank line
      -- (or the end of the file)
      line = file:read("*line")
      local entry = ""
      while (line~=nil) and (line~="") do
         entry = entry .. line
         line = file:read("*line")
      end
      -- remove the final dot (if present)
      local entryLen = string.len(entry)
      local lastChar = string.sub(entry, entryLen, entryLen)
      if lastChar == "." then
        entry = string.sub(entry, 1, entryLen-1)
      end
      entries[key]=entry
    end
    line = file:read("*line")
  end
  file:close()
end

-- looks up the given key in the entries
-- in case an entry is not found, a bold question mark is printed
function bibentry(key)
  local res = entries[key[1]]
  if res==nil then
     res = "\\textbf{?}"
  end
  tex.print(res)
end

```

6 Implementation of LaTeX Package `luabibentry.sty`

LuaLaTeX must be used to use the package.

```
\RequirePackage{ifluatex}
\ifluatex\else
  \PackageError{luabibentry}{lualatex needed}{%
    Package ‘luabibentry’ needs LuaTeX.\MessageBreak
    So you should use ‘lualatex’ to process you document!\MessageBreak
    See documentation of ‘luabibentry’ for further information.}%
  \expandafter\expandafter\expandafter\csname endinput\endcsname
\fi
```

Load the lua module:

```
\directlua{require("luabibentry.lua")}
```

Interface to the lua module:

```
\newcommand{\setupbibentry}[1]{\directlua{luabibentry.builddata{"#1.bbl"}}}
\newcommand{\bibentry}[1]{\nocite{#1}\directlua{luabibentry.bibentry{"#1"}}}
```

7 Acknowledgements

This package is a variant of `bibentry.sty` by Patrick W. Daly. `bibentry.sty` is distributed with the `natbib` package. This documentation of luabibentry is mostly adapted from Patrick’s documentation of `bibentry`.

Thanks to Markus Kohm for the dtx and lua inspirements, Manuel Pégourié-Gonnard for the Makefile, dtx, and lua inspirements, and Heiko Oberdiek for his detailed feedback on a draft version of this package.