biber

A backend bibliography processor for biblatex

François Charette, Philip Kime
firmicus@ankabut.net,
Philip@kime.org.uk

Version 1.2
28th September 2010

Contents

1 Introduction

1.1 About

`biber` is conceptually a `bibtex` replacement for `biblatex`. It is written in Perl with the aim of providing a customised and sophisticated data preparation backend for `biblatex`. Functionally, it offers a superset of `bibtex`'s capabilities but is tightly coupled with `biblatex` and cannot be used as a stand-alone tool with standard `.bst` styles.

1.2 Requirements

`biber` is distributed in two ways. There is a perl source version which requires you to have a working perl installation (preferably version 5.12 but no less than 5.10) and the ability to install the pre-requisite modules. Also provided are binaries for major OSes built with the perl `PAR::Packer` module and utilities.

Currently there are binaries available for:

- OSX Intel 64-bit
- Windows
- Linux 32-bit
- Linux 64-bit

These should work on any fairly recent OS version. Both binaries and perl source are available on SourceForge[1].

1.3 License

`biber` is released under the free software Artistic License 2.0[2]

1.4 History

`bibtex` has been the default (only ...) integrated choice for bibliography processing in TeX for a long time. It has well known limitations which stem from its data format, data model and lack of UTF-8 support[3]. The `.bst` language for writing bibliography styles is painful to learn and use. It is not a general programming language and this make it really very hard to do sophisticated automated processing of bibliographies.

`biblatex` was a major advance for LaTeX users as it moved much of the bibliography processing into LaTeX macros. However, `biblatex` still used `bibtex` as a sorting engine for the bibliography and also to generate various labels for entries. `bibtex`'s capabilities even for this reduced set of tasks was still quite restricted due to the lack of UTF-8 support and the more and more complex programming issues involved in label preparation and file encoding.

`biber` was designed specifically for `biblatex` in order to provide a powerful backend engine which could deal with any required tasks to do with `.bbl` preparation. It can

- Deal with the full range of UTF-8
- Sort in a completely customisable manner, using when available, CLDR collation tailorings
- Automatically encode the `.bbl` into any supported encoding format[4]
- Process all bibliography sections in one pass of the tool
- Handle UTF-8 citekeys and filenames (given a suitable fully UTF-8 compliant TeX engine)
- Handle very complex auto-expansion and contraction of names and namelists.
- Lots of other things

[1] `http://sourceforge.net/projects/biblatex-biber/`
[2] `http://www.opensource.org/licenses/artistic-license-2.0.php`
[3] In fact, there is now a UTF-8 compliant version
[4] 'Supported' here means encodings supported by the Perl `Encode` module

1.5 Acknowledgments

François Charette originally wrote `biber`. Philip Kime joined in the development in 2009.

2 Use

Firstly, running `biber --help` will display all options and a brief description of each. This is the most useful brief source of usage information.

With the `backend=biber` option, `biblatex` switches its backend interface and passes all options and information relevant to `biber`'s operation in a control file with extension `.bcf`[5]. This is conceptually equivalent to the `.aux` file which LaTeX uses to pass information to `bibtex`. The `.bcf` file is XML and contains many options and settings which configure how `biber` is to process the bibliography and generate the `.bbl` file.

The usual way to call `biber` is simply with the `.bcf` file as the only argument. The '`.bcf`' extension of the control file is not optional. `biblatex` always outputs a control file with the `.bcf extension`. Specifying the '`.bcf`' extension to `biber` *is* optional. Assuming a control file called `test.bcf`, the following two commands are equivalent:

```
biber test.bcf
biber test
```

2.1 Options and config file

`biber` sets its options using the following resource chain which is given in decreasing precedence order:

command line options →
  `biber.conf` file →
    `.bcf` file→
      `biber` hard-coded defaults

Users do not need to care directly about the contents or format of the `.bcf` file as this is generated from the options which they specify for `biblatex`. To override the `.bcf` options or to provide option settings when no `.bcf` file is used (see for example the `--allentries|-a` and `--bibdata|-d` options), users may use either a configuration file or the command line to set options.

[5] BibLaTeX Control File

The configuration file is by default called `biber.conf` but this can be changed using the `--configfile|-g` option. Unless `--configfile|-g` is used, the config file is looked for in the following places, in decreasing order of preference:

`biber.conf` in the current directory →
  `$HOME/.biber.conf` →
    `$ENV{XDG_CONFIG_HOME}/biber/biber.conf` →
      `$HOME/Library/biber/biber.conf` (Mac OSX only) →
      `$ENV{APPDATA}/biber.conf` (Windows only) →
        the output of `kpsewhich biber.conf` (if available on the system)

The config file format is a very flexible one which allows users to specify options in most common formats, even mixed in the same file. It's easier to see an example. Here is a config file which displays the `biber` hard-coded defaults:

```
allentries          0
bblencoding         UTF-8
bibdata             undef
bibending           UTF-8
collate             1
<collate_options>
    level           2
</collate_options>
debug               0
fastsort            0
mincrossrefs        2
nolog               0
nosortdiacritics    [\x{2bf}\x{2018}]
nosortprefix        \p{L}{2}\p{Pd}
quiet               0
sortcase            1
sortlocale          en_US.utf8
sortupper           1
trace               0
validate            0
wraplines           0
```

You can see here that options with multiple key/value pairs of their own like `--collate_options|-c` can be specified in Apache config format. Please see the documentation for the `Config::General` Perl module[6] if you really need details.

[6] http://search.cpan.org/search?query=Config::General&mode=all

In practise, if you use a config file at all for `biber`, it will contain very little as you will usually set all options by setting options in `biblatex` which will pass them to `biber` via the `.bcf` file.

The `--collate_options|-c` option takes a number of key/value pairs as value. See section 2.4 for details. The value of the `nosort*` options can only be set in the config file and not on the command line. This is because the values are Perl regular expressions and would need special quoting to set on the command line. This can get a bit tricky on some OSes (like Windows) so it's safer to set then in the config file. They specify stand-alone diacritic marks and name prefices to strip before sorting takes place and are designed to deal with cases like

```
author     = {{al-Hasan}, ʿAlī},
```

where the prefix 'al-' and the diacritic '^{c}' should not be considered when sorting.

2.2 Input/Output File Locations

2.2.1 Control file

The control file is normally passed as the only argument to biber (unless using `--allentries|-a` and `--bibdata|-d`). It is searched for using the following locations, in decreasing order of priority:

Absolute filename →
  In the `--output-directory`, if specified→
    Relative to current directory→
      Using `kpsewhich`, if available

2.2.2 Database files

Bibliography database files are searched for using the same rule as for control files (see section 2.2.1 above). Unless using the `--bibdata|-d` option, users usually do not specify explicitly the bibliography database files; they are normally passed in the `.bcf` control file, taken from the `biblatex \bibliography{}` command arguments.

2.3 Logfile

By default, the logfile for biber will be name `\jobname.blg`, so, if you run

```
biber <options> test.bcf
```

then the logfile will be called 'test.blg'. Like the .bbl output file, it will be created in the --output-directory|-c, if this option is defined. If there is no .bcf file on the command line (for example, when using the --allentries|-a and --bibdata|-d options), then the logfile name will default to 'biber.blg'. You can override the logfile name by using the --logfile option:

```
biber --logfile=lfname test.bcf
```

results in a logfile called 'lfname.blg'.

2.4 Collation and Localisation

biber takes care of collating the bibliography for biblatex. It writes entries to the .bbl file sorted by a completely customisable set of rules which are passed in the .bcf file by biblatex. biber has two ways of performing collation:

--collate|-C
: The default. This option makes biber use the Unicode::Collate module for collation which implements the full UCA (Unicode Collation Algorithm). It also has CLDR (Common Locale Data Repository) tailoring to deal with cases which are not covered by the UCA. It is a little slower than --fastsort|-f but the advantages are such that it's rarely worth using --fastsort|-f

--fastsort|-f
: Biber will sort using the OS locale collation tables. The drawback for this method is that special collation tailoring for various languages are not implemented in the collation tables for many OSes. For example, few OSes correctly sort 'å' before 'ä' in the Swedish (sv_SE) locale. If you are using a common latin alphabet, then this is probably not a problem for you.

The locale used for collation is determined by the following resource chain which is given in decreasing precedence order:

--collate_options|-c (e.g. -c 'locale => "de_DE"') →
  --sortlocale|-l →
    LC_COLLATE environment variable →
      LANG environment variable →
        LC_ALL environment variable

With the default `--collate|-C` option, the locale will be used to look for a collation tailoring for that locale. It will generate an information warning if it finds none. This is not a problem as the vast majority of collation cases are covered by the basic standard UCA and many locales neither have nor need any special collation tailoring.

With the `--fastsort|-f` option, the locale will be used to locate an OS locale definition to use for the collation. This may or may not be correctly tailored, depending on the locale and the OS.

Collation is by default case sensitive. You can turn this off using the `biber` option `--sortcase=0` or from `biblatex` using its option `sortcase=false`.

`--collate|-C` by default collates uppercase before lower. You can reverse this using the `biber` option `--sortupper=0` or from `biblatex` by using its option `sortupper=false`.

There are in fact many options to `Unicode::Collate` which can tailor the collation in various ways in addition to the locale tailoring which is automatically performed. Users should see the the documentation to the module for the various options, most of which the vast majority of users will never need[7]. Options are passed using the `--collate_options|-c` option as a single quoted string, each option separated by comma, each key and value separated by '=>'. See examples.

2.4.1 Examples

`biber`
: Call biber using all settings from the `.bcf` generated from the LaTeX run. Case sensitive UCA sorting is performed taking the locale for tailoring from the environment if no `sortlocale` is defined in the `.bcf`

`biber --sortlocale=de_DE`
: Override any locale setting in the `.bcf` or the environment.

`biber --fastsort`
: Use slightly quicker internal sorting routine. This uses the OS locale files which may or may not be accurate.

`biber --sortcase=0`
: Case insensitive sorting.

`biber --sortupper=0 --collate_options="backwards => 2"`
: Collate lowercase before upper and collate French accents in reverse order at UCA level 2.

[7] For details on the various options, see http://search.cpan.org/search?query=Unicode%3A%3ACollate&mode=all

2.5 Encoding of files

biber takes care of reencoding the .bib data as necessary. In normal use, biblatex passes its bibencoding option value to biber via the .bcf file. It also passes an option bblencoding the value of which is derived from the inputenc package setting (if the user is using this), otherwise 'utf8' (for XeTeX or LuaTeX) or 'ascii' (any other TeX engine).
biber performs the following tasks:

1. Decodes the .bib into UTF-8 if it is not UTF-8 already
2. Decodes LaTeX character macros into UTF-8
3. Encodes the output so that the .bbl is in the encoding that bblencoding specifies
4. Warns if it is asked to output to the .bbl any UTF-8 decoded LaTeX character macros which are not in the bblencoding encoding. Replaces with a diacritic-stripped substitute

As you can see from item 2 above, by default, biber converts LaTeX character macros into UTF-8 internally. This is very useful as it means that things are sorted correctly but has two potential (but rare) problems which you should be aware of:

- If you are using PDFLaTeX and \usepackage[utf8]{inputenc}, it is possible that the UTF-8 characters resulting from biber's internal LaTeX character macro decoding break inputenc. This is because inputenc does not implement all of UTF-8, only a commonly used subset.
  An example–if you had \DJ in your .bib, biber decodes this correctly to 'Đ' and this breaks inputenc because it doesn't understand that UTF-8 character. The solution here is to switch to a TeX engine with full UTF-8 support like XeTeX or LuaTeX as these don't use or need inputenc.
- If your bblencoding is not UTF-8, and you are using some UTF-8 equivalent LaTeX character macros in your .bib, then some .bbl fields (currently only \sortinit{}) might end up with invalid characters in them, according to the .bbl encoding. This is because some fields must be generated from the final sorting data which is only available after the LaTeX character macro decoding step.
  For example, suppose you were using PDFLaTeX with \usepackage[latin1]{inputenc} and the following .bib entry

```
@BOOK{citekey1,
  AUTHOR = {{\v S}imple, Simon},
}
```

With normal LaTeX character macro decoding, the `{\v S}` is decoded into 'Š' and so with name-first sorting, `\sortinit{}` would be 'Š'. This is an invalid character in latin1 encoding and so the `.bbl` would be broken. In such cases when `\sortinit{}` is a char not valid in the `bblencoding`, `biber` strips off any diacritics which in this case results in 'S'. This is not ideal as this is not the initial character of the string used for sorting any more but it's a decent replacement in such cases. The solution is really to use UTF-8 `bblencoding` wherever possible. In extreme cases like the one above, this might also mean switching TeX engines to one that supports full UTF-8.

Normally, you do not need to set the encoding options on the `biber` command line as they are passed in the `.bcf` via the information in your `biblatex` environment. However, you can override the `.bcf` settings with the command line or config file. The resource chain for encoding settings is, in decreasing order of preference:

`--bibencoding|-e` and `--bblencoding|-E` →
  `biber` config file →
    `.bcf` control file

2.5.1 Examples

`biber`
: Read `bibencoding` and `bblencoding` from the config file or `.bcf`.

`biber --bblencoding=latin2`
: Encode the `.bbl` as latin2, overriding the `.bcf`.

`biber -u`
: Shortcut alias for `biber --bibencoding=UTF-8`

`biber -U`
: Shortcut alias for `biber --bblencoding=UTF-8`

2.6 Limitations

Currently, users are restricted to the bibliography entry types hard-coded into `biblatex`. This is mitigated a little by the custom fields listed in section 2.2.4 of the `bibatex` manual but these are not portable or semantically obvious in their meaning. It is planned to have a customisable interface in `biblatex` which will allow users to define entry types an fields and have these passed through to `biber` which will validate the structure of the bibliography against these definitions. This would allow a fully customisable data model interface. Currently this is impossible due to a reliance on the `.bib` format which is quite restricted in scope and extensibility. It is likely that `biblatex` and `biber` will move to a modular data layer with an XML format as the default. `.bib` support will be maintained as a legacy format.

Currently it is not possible to automatically expand name lists to their minimally unique truncation which is required by some styles (APA for example). This is quite a hard problem which is implemented in an experimental `biber` branch but need `biblatex` support, envisaged for version 2.x. It requires an enhanced `.bbl` format, among other things.