Bib2x

for processing BibTeX-bibliographies

Alexander Feder
i@xandi.eu

28. 4. 2006

Problems

- Ever growing collection of bibliographic references
  - which quickly becomes a mess
- Properly organizing BIBTEX-libraries
  - tedious and time-consuming when done by hand
  - always out-of-date
  - error-prone
- Using the bibliographies outside of LATEX
  - Conversion to
    - other bibliography formats
    - XML, Plaintext etc. for use in 3^{rd}-party applications
  - Transformation to (e.g.) HTML for presentation on the web
- Pretty Printing

Additional features demanded

- Mechanism to
  - Filter
  - Sort
- Integrability
  - Shell
  - Cron-Jobs
- Platform-Independence
- Applicable on large databases

Bib2x

Bib2x

- Developed in C++
- using Lex and Yacc for parsing
- using ODB for internal representation

BIB2X

What does BIB2x do?

- Reads BIBTEX-libraries from `stdin` or a specified file
- Outputs
  - built-in formats: html, plaintext, odb-dump
  - arbitrary formats using templates
    - proof of concept: html, rtf, BIBTEX
- allows to filter and sort by any regular BIBTEX-field...
- ...and *any* self-defined field

Bib2x

What is meant with *„self-defined" field*?

```
@string  { pbl:prock = "Prockhouse" }

@article { feder05,
           author    = "Alexander Feder",
           title     = "Experimental usage of the {ODB} for " #
                       "multipurpose manipulation of and " #
                       "access to Bibtex-Bibliographies",
           journal   = "Prawda",
           pages     = "51-82",
           publisher = pbl:prock,
           year      = 2005,
           category  = "pure nonsense" # "selfmade"
           }
```

BIB2X

What does BIB2x do? (cont.)

- Converts special characters into UTF
  - ö: \"{o} → ö
- repertoire of understood LaTeX-commands
  - extensible (but a little bit pedestrian)

Templates

Templates are processed onto the entire set of BIBTEX-entries which have been parsed.

Templates define

- The format of the output file
- Which entries are processed...
  - Exclusions
  - Inclusions
- ...and how they are processed
  - Grouping
  - Sorting

Looping, Grouping, Reducing

%%GRP%% ... %%PRG%%

Split input-set into multiple sets which are then looped through one subset after the other.

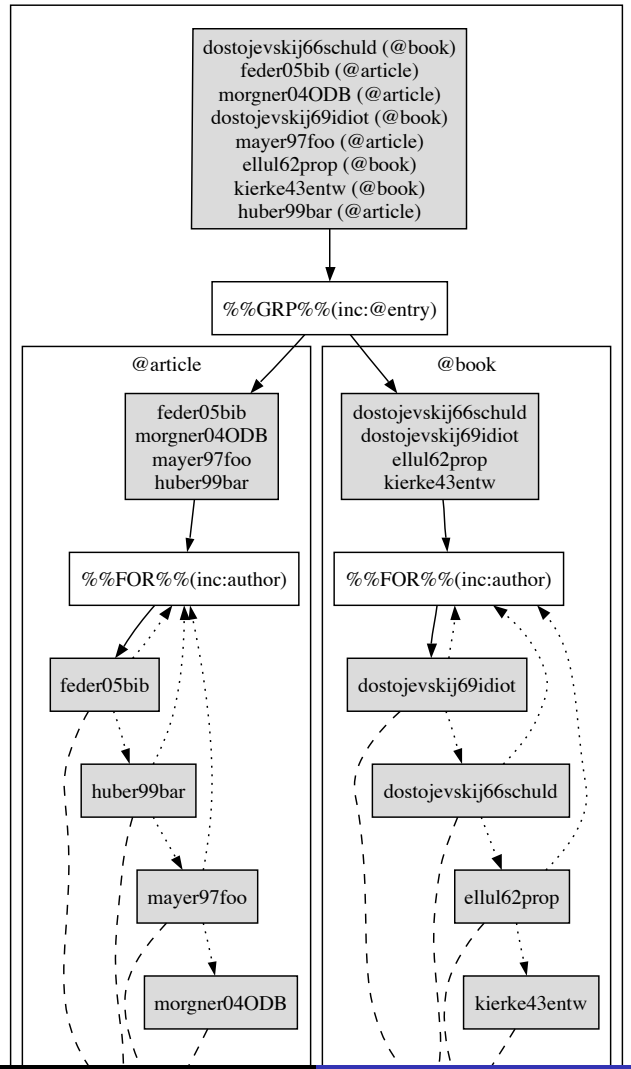
%%FOR%% ... %%FOR%%

Loops through the current set, one bibtex-entry after the other.

%%RED%%

Reduce the current set using the specified constraints.

Looping, Grouping and Reducing (cont.)



Looping, Grouping and Reducing (cont.)

CMD $\in$ { GRP, FOR, RED }

%%CMD%%

%%CMD%% (Sort: Condition; Constraint; Constraint; ..;)
Sort $\in$ { inc, dec }

Looping, Grouping and Reducing (cont.)

Condition

- name of the tag to be used as key (e.g. `author`)
- `@entry` to specify that sorting should follow the BIBTEX-entry-type.

Looping, Grouping and Reducing (cont.)

Condition

```
tag:nr op {"foo", "bar",... }
```

`tag` the tag to be operated on

`:nr` (optional) specifies which component of the tag is used

- `0` operation will perform on concatenation of all substrings
- `x` each substring is operated on separately
- `k` ($k \in \mathbb{N}$) operation will perform on k^{th} substring

`op` operator

- `=` tag contains at least one of the specified words.
- `!=` tag does not contain any of the specified words.

Looping, Grouping and Reducing (cont.)

```
%%FOR%% ( dec: author; author = {
"alexander" }; author!={"bob","john"}; )
%%FOR%% ( inc: category:2 ={"selfmade",
"nonsense"}; title!={"interesting"}; )
```

Note

- a set of words enclosed by braces equals logical `OR`
- subsequent constraints equal logical `AND`

Group

%%GRP%%

GROUP splits the previous set into subsets which then are separately processed according to the body of the GROUP.

Commands valid in %%GRP%%'s Body:

`%%name%%` returns the name of the current subset

`%%FOR%%` allows processing the entries inside this subset one by one.

`%%GRP%%` to construct arbitrarily nested Groups

Note

- a set of words enclosed by braces equals logical `OR`
- subsequent constraints equal logical `AND`

Example Templates

Example Template using %%GRP%%

```
<html><body>
  %%GRP%%(inc: author)<div style="...">
    Works by <b>%%name%%</b> were published in
    <ul>
      %%GRP%%(inc:year)
      <li style="color:blue">%%name%%</li>
      %%PRG%%
    </ul></div><br>
  %%PRG%%
</body></html>
```

Example Templates - Result

Execution of BIB2x

```
$ bib2x -f test.bib -t temp.late
```

```
<html><body>
 <div style="...">
   Works by <b>Alexander Feder</b> were published in
   <ul>
     <li style="color:blue">2004</li>
     <li style="color:blue">2005</li>
   </ul></div><br>
 <div style="...">
```

For

%%FOR%%

FOR iterates through the current set of entries, and processes each entry according to the body of the FOR.

Commands valid in %%FOR%%'s Body:

`%%$odbid%%`	returns the ODB-Id of the current entry.
`%%$bibkey%%`	returns the BIBTEX-citation key.
`%$bibtype%%`	returns the BIBTEX-entry type
`%%anytag%%`	where anytag is any tag like `author` or `year`. Returns the concatenated content of the corresponding entry
`%%BIB%%`	enables conditioned access to all tags without prior knowledge of their existence.

Bib

%%BIB%%

`%%BIB%%` It iterates through all tags and, by default, returns them all. This can be conditioned and filtered. It is used to

- process tags previously unknown and
- cope with tags expected but not existing.
- specify the order in which these tags shall be processed and
- define under which conditions tags may be skipped or uniquely included.

Beware!

The syntax of `%%BIB%%` is different to the syntax of `%%FOR%%`, but admittedly confusingly similar!

Bib

%%BIB%%

```
%%BIB%%(Sort: Sequence;Constraint;Constraint;...;)
```

`Sequence` is a comma-separated list of tags with prefix:

- `!` excludes a tag. No operation will be performed
- `#` makes the tag(s) exclusive.

  defines the order

Bib

%%BIB%%-Body

```
%%BIB%%(Sort: Sequence;Constraint;Constraint;...;)
```

X_1=" ... "

⋮

X_i=" ... "

```
%%BIB%%
```

$X_i \in \{G,M,E,N\}$, $1 \leq i \leq 4$

- M : MISSING – in case a *specified* tag has not been found.
- E : EMPTY – in case the specified tag has been found empty.
- N : NONEMPTY – for specified tags having content.
- G : GENERAL – for *non*-specified tags having content.

Bib

Commands valid inside the string of the parts return

%%name%%	the name of the current tag.
%%content%%	a concatenation of the tag's substrings.
%%step:#-#%%	a range of the tag's content
%%MOD_XX%%	for modifying a returned string or substring.

Step

%%step:#-#%% ("Prefix" ; "Middle" ; "Suffix")

\# is

- \# numeric value or
- x the letter „x" defining everything until the end

Example

Example:

```
%%BIB%%(inc:#category;)
N="%%name%%: <a
href=\"http://host.at/query?key=%%content%%\">%%content%%</a>"
M="<span style=\color:red\">Categorization
missing!</span>"
E="Not member of any category"
%%BIB%%
```

Modifications

%%MOD_XX%%

```
%%MOD_XX%%  (  %%CMD%%  )
```

where XX is:

- LC LOWERCASE – everything is converted to lower-case letters.
- LF LOWERCASE, FIRSTUPPER – everything is converted to lower-case letters except the very first one.
- UC UPPERCASE – everything is converted to upper-case letters.
- UB UPPERCASE BEGIN – every word starts with an upper-case letter.

Example

%%MOD_XX%%

%%MOD_LC%% (”A very interesting Text about {BiBTeX{”)
%%MOD_LF%% (”A very interesting Text about {BiBTeX{”)
%%MOD_UC%% (”A very interesting Text about {BiBTeX{”)
%%MOD_UB%% (”A very interesting Text about {BiBTeX{”)

Example

%%MOD_XX%% Result

a very interesting text about BiBTeX
A very interesting text about BiBTeX
A VERY INTERESTING TEXT ABOUT BiBTeX
A Very Interesting Text About BiBTeX

Danke

Danke!

Looping, Grouping and Reducing (cont.)