KBSET/Letters – User Guide

Christoph Wernhard

Draft – October 11, 2020

Contents

1 Introduction **3**

2 Prerequisites **4**
2.1 Required Further Software 4
2.2 Supported Platforms 4

3 Notation and Shorthands Used in this Document **5**

4 Installation **5**

5 Typical Set-Up and Overview on Files and Directories **6**

6 LaTeX Set-Up in Main Files **6**
6.1 Toggles 6
6.2 Indexes 8
6.3 Further Redefinable Commands 8

7 Configuration **8**
7.1 The *KBSET/Letters* Configuration File 8
7.2 File and Directory Specifiers 9
7.3 Configuration Keys 9

8 Shell Scripts **11**
8.1 General Remarks 11
8.2 Scripts 12

9 Usage Scenarios **13**
9.1 General Notes 13
9.2 PDF Generation During Development 14
9.3 PDF Generation 14
9.4 HTML Generation 14

10 Consistency Validation **14**
10.1 General Remarks 15
10.2 Basic Validation Beyond the LaTeX Standard Workflow 15
10.3 Validation on the Basis of LaTeX Log Files 15
10.4 Validation of the Bibliography 16
10.5 Further Possibilities 16

11 Using the System from Prolog **16**
11.1 Overview on Relevant Modules 16
11.2 Conversion to Further Formats 17

1 Introduction

KBSET/Letters is an environment for developing scholarly editions of correspondences. It is realized as a specific application pattern of the more general environment *KBSET (**K**nowledge-**B**ased **S**upport for Scholarly **E**diting and **T**ext Processing). KBSET/Letters* is successfully applied in a comprehensive scholarly edition project on a correspondence from the 18th century.[1]

At its current stage, the *KBSET/Letters* environment is fully adequate for scholarly print editions of correspondences from the 18th and 19th century that are in German language and where the edited texts are represented in a character-preserving (*zeichengetreu*) but not position-preserving (*positionsgetreu*) way.[2] The implementation of HTML presentations is still under development, but already sufficient to generate draft versions of online editions with fairly high quality. The main characteristics and features of *KBSET/Letters* are as follows:

- *KBSET/Letters* supports the *complete workflow* of creating a scholarly edition, from creating transcriptions, annotations and data bases with meta information, via intermediate representations for revision and consistency checking, to high-quality PDF presentations for printing and online viewing as well as high quality HTML presentations.
- The environment is entirely based on *free software*. Aside of the *KBSET* core system, these are widespread platform-independent software packages: a TEX distribution and the *SWI-Prolog* programming system.
- *All that is required to reproduce* the generation of representative high-quality PDF and HTML presentations from the source documents of an edition project can be published as free documentation or free software, respectively, and thus be considered as a component of the edition. This ensures a high degree of sustainability and facilitates the use of the source documents in further contexts by inspecting, modifying or extending the involved free software.
- A *parsimonious set of declarative markup elements* (*KBSET/Letters Markup (KLM)*) has been specified that is adapted to the requirements of scholarly editions of correspondences from the 18th and 19th century. Working with such specialized markup elements is perceived by users as expressing statements of interest rather than a technical burden.
- The *markup is expressed in LATEX*, as commands and environments. Text with markup thus remains fairly readable and can be directly created by users with

[1] The correspondence of Johann Georg Sulzer (1720–1779) and Johann Jakob Bodmer (1698–1783), transcribed and annotated by J. Kittelmann with assistance of B. Baumann, which is going be published in 2020 in print as volume 10 of Sulzer's *Gesammelte Schriften*, edited by H. Adler and E. Décultot.

[2] Since the represented text is encoded in UTF-8, the limitations with respect to language are easy to overcome by revisiting the handling of text phrases used to generate presentations and replacing the used BibLaTeX configuration.

any text editor that supports LaTeX. Moreover, a way to process the specialized markup has been *implemented in LaTeX* to produce PDF presentations of fair quality. In addition, a second way of processing has been implemented, where the LaTeX sources are parsed and converted to a high-quality HTML presentation. The parser can also be used for preprocessing the marked-up LaTeX sources, for example to re-arrange letters that are maintained in different source documents according to their author into the temporal order of a correspondence, to obtain PDF documents of high quality.

- The core system of *KBSET/Letters* is *written in SWI-Prolog*, which includes good interfaces to XML and RDF and realizes the potential of Prolog as a unifying language. As noted on the *SWI-Prolog* home page,[3] it considers Prolog *"primarily as glue between various components. The main reason for this is that data is at the core of many modern applications while there is a large variety in which data is structured and stored. Classical query languages such as SQL, SPARQL, XPATH, etc. can each deal with one such format only, while Prolog can provide a concise and natural query language for each of these formats that can either be executed directly or be compiled into dedicated query language expressions. Prolog's relational paradigm fits well with tabular data (RDBMS), while optimized support for recursive code fits well with tree and graph shaped data (RDF)."* As an *AI programming language*, Prolog already includes human readable data serialization, which, for other programming languages is typically supplied with XML. The use of Prolog in *KBSET* provides a basis for reaching out into *semantics-based* techniques of knowledge representation and knowledge processing.

2 Prerequisites

2.1 Required Further Software

The environment requires the *TeX Live* TeX distribution and *SWI-Prolog*, which are both free software. A terminal program to invoke shell commands and *Bash* scripts as well as a text editor with LaTeX support (e.g., *GNU Emacs*) are presupposed.

The environment has been tested with *TeX Live 2018* and *SWI-Prolog 8.0.3*. Other comparable TeX distributions as well as older versions of *SWI-Prolog* (e.g., versions 7.X) might also work.

2.2 Supported Platforms

The environment should work on all Unix-like platforms like Linux and OS/X.

On Microsoft Windows it works with *Cygwin* with the included *SWI-Prolog* package `pl-7.2.3-1` and TeX Live 2018. It has been tested there with an installation of TeX Live 2018 that is independent from *Cygwin* (i.e., not as *Cygwin* package) and made

[3] `https://www.swi-prolog.org/features.html`, accessed Nov 21 2019.

accessible from the *Cygwin* shell by adding its directory with binaries to PATH (e.g., in `.bash_profile`).

3 Notation and Shorthands Used in this Document

- [*KLM*]: *KBSET/Letters Markup*, as specified in Jana Kittelmann and Christoph Wernhard: *KBSET/Letters Markup (KLM): Parsimonious Descriptive Markup for Scholarly Editions of Correspondences*.[4].
- `KBSET`: The root directory of the *KBSET* distribution, by default called `kbset`.
- `DOCDIR`: The user's working directory for the scholarly edition project.
- `$`: Indicates a shell command invocation.

4 Installation

- *KBSET* is distributed as a directory, which is either generated via `git` or created by unpacking a TAR archive. (`git` facilitates updating that directory with new versions of the distribution.) That directory, referred to as `KBSET` in this documentation, can be placed anywhere on the system. It should be writable for the user because some larger Prolog source files (e.g., with lists of words) are automatically compiled to SWI-Prolog's quick load format when they are loaded for the first time. The compiled files are stored at the same place as the sources.
- The directory `KBSET/bin` should be added to the command search path, for example by a line

  `PATH=$PATH:KBSET/bin`

  in `~/.profile` (or `~/.bash_profile` if there is no file `~/.profile`), where `KBSET` stands for the path name of the `KBSET` directory.
- It must be ensured that `pdflatex` finds the content of `latex/kbset`. This can be achieved, for example, by linking or copying `KBSET/latex/kbset` to

  `TEXMFHOME/tex/latex/kbset`,

  where `TEXMFHOME` stands for the value of:

  ```
  $ kpsexpand '$TEXMFHOME'
  ```

  The success of the installation can be checked by:

  ```
  $ kpsewhich -progname=pdflatex kbsetletters.sty
  ```

[4] Available from `http://cs.christophwernhard.com/kbset`

5 Typical Set-Up and Overview on Files and Directories

A typical `DOCDIR` directory, that is a *KBSET/Letters* working directory for a scholarly edition project, contains files and directories as discussed below. (Of course, `DOCDIR` must be writable for the user, since the LaTeX workflow and the *KBSET* core processor place their outputs there.) For an example see

`KBSET/example/letters_mini`

1. Two LaTeX main files. In the example, these are `main.tex` and `main_devel.tex`. The first represents the final document, the second an intermediate document that is useful for developing the scholarly edition and can be processed just by a LaTeX workflow, without invoking the *KBSET* core processor.

   Both main files include at processing further LaTeX files. A difference between them is that the first includes files that will be generated by the *KBSET* processor, while the second one only files that are supplied by the user of included with the *KBSET* system. Main files contain the technical setup of the LaTeX environment for *KBSET/Letters*. User options are described in Sect. 6 below.

2. LaTeX files with letters and with annotations, that is, with `letter` and `annotation` environments (see [*KLM*]). In the example there is just a single letters and a single annotation file `letters.tex`, and `annotations.tex`, respectively.

3. LaTeX files with fact bases (see [*KLM*]). In the example there is only one such file: `factbase.tex`.

4. Further LaTeX files that are included in the main files. In the example, these are `specials.tex` (some hyphenation rules for 18th century German) and `appendix.tex` (writing of bibliography and indexes).

5. Further directories. In the example: `bib`, `extras_html`, `data` and `gendata`. See Sect. 7.3 below.

6. A file `kbset_config.pl` with the configuration of the *KBSET* core processor for the project. See Sect. 7 below.

6 LaTeX Set-Up in Main Files

Main files contain the technical setup of the LaTeX environment for *KBSET/Letters*, as shown in the example files

`KBSET/example/letters_mini/main.tex` and
`KBSET/example/letters_mini/main_devel.tex`

6.1 Toggles

Some *KBSET/Letters*-specific settings can be specified by LaTeX toggles in main files.

[*toggle*] `useminion`
: Use the MinionPro font, if it is installed. Default: *false*.

[*toggle*] `minionlooseqoutes`
: If MinionPro is used, load it with the `loosequotes` option. Default: *false*.

[*toggle*] `usefrutiger`
: Use the Frutiger font as sans-serif, if it is installed. Default: *false*.

[*toggle*] `histcurrencysymbols`
: Use images for historic currency symbols. The images need to be installed separately. Default: *false*.

[*toggle*] `linerefsversal`
: In page/line references in annotations: Use lining figures (*Versalziffern*) and subscript instead of non-lining figures (*Mediävalziffern*) with superscript. Set automatically to *true* if MinionPro is used. Default: *false*.

[*toggle*] `marginmarks`
: Indicate text places in letters which have an associated annotation with an asterisk on the margin. Requires consideration in the `geometry` options, for example by `reversemarginpar` or by `marginparsep=16pt,marginparwidth=10pt`. Default: *false*.

[*toggle*] `inversevideo`
: Inform the system that the representation is bright text on dark ground (the toggle alone does not effect the color change). Default: *false*.

[*toggle*] `swissquotes`
: Use French quotes with Swiss instead of German (and Austrian) style: «Swiss», ‹Swiss›, »German«, ›German‹. Default: *false*.

[*toggle*] `showundefs`
: Show undefined entities in indexes. (Also if the toggle is not set, the undefined entities can be determined by `grep IDX` from the `.log` files.) Default: *false*.

[*toggle*] `printxmeta`
: Print out information declared with `\xmeta` statements. The information is printed at the beginning of the respective annotations. Default: *true*.

[*toggle*] `showids`
: [Not maintained] Show symbolic letter identifiers in headers of letters and annotations (for debugging). Default: *false*.

[*toggle*] `showqrcode`
: [Not maintained] Print a QRCode with the letter URL in the header of an annotation. An experimental way of linking printed and digital formats. Default: *false*.

[*toggle*] `debugmeta`
: [Not maintained] Highlight meta information (for debugging), e.g., by colors. Default: *false*.

[*toggle*] `debugmargin`
: [Not maintained] Show meta information in the margin (for debugging). Default: *false*.

6.2 Indexes

At LaTeX processing with the `kbsetletters` style several indexes for xindy are created. Headers for them are defined in the `kbsetlettersinit` style file such that they can be redefined with `\renewcommand` after loading that style file and re-used for printing out the indexes, as shown in `appendix.tex` in the example.

Index	Title Command	Indexed Items
`per`	`\idxnamePer`	Persons and works
`cor`	`\idxnameCor`	Corporations, institutions
`jou`	`\idxnameJou`	Journals
`geo`	`\idxnameGeo`	Geographical locations
`dat`	`\idxnameDat`	Date specifications
`eve`	`\idxnameEve`	Events
`sub`	`\idxnameSub`	Subjects, concepts

6.3 Further Redefinable Commands

`\bibliographyTitle` – main title of the bibliography, used similarly to `\bibname`. `\annotationChapterTitl` – title of the annotation chapter, used in page headers. `\pageformat` – determines a file with page format settings.

7 Configuration

7.1 The *KBSET/Letters* Configuration File

The complete project specific configuration of the *KBSET/Letters* core tools is in a single configuration file that is written in Prolog syntax, for example

`KBSET/example/letters_mini/kbset_config.pl`

and loaded into *SWI-Prolog* after loading the *KBSET/Letters* core system.

The shell scripts that come with *KBSET* automatically load such a configuration file, as specified in Sect. 8.

7.2 File and Directory Specifiers

Note on Prolog Syntax. Configuration files are written in Prolog syntax. For users unfamiliar with that syntax, here are a few notes:

In the configuration files, file and directory specifiers are represented by Prolog symbols (technically called *atoms*). These can contain arbitrary characters and in general have to be written in single quotes. If the first character is a lowercase letter and all other characters are alphanumeric or the underscore `_`, then the single quotes can be omitted. The empty atom is denoted by `''`.

Certain configuration keys can have a set of values, represented by a Prolog *list*. A list is written in square brackets `[` and `]`. Its elements are separated by commas. The empty list is written as `[]`. As a special case we use lists of *key*=*value* pairs

As the Prolog syntax is term-oriented, white space can be used arbitrarily to facilitate readability.

Determining `DOCDIR`. The directory `DOCDIR` is determined as follows:

- If *SWI-Prolog* was invoked with an application command line argument

  `-docdir=`*DocDir*,

  then *DocDir* is taken.
- Else, if the environment variable `KBSET_DOCDIR` is bound, its value is taken.
- Else, the directory containing the file in which the predicate `kbset_config`/2 was defined (typically `kbset_config.pl`) is taken.

Interpretation of File and Directory Specifiers. A path name specifier (i.e., a specifier of a file or directory) is interpreted as follows:

- If the specifier begins with `/`, then it is taken as an absolute filename.
- If the specifier is of the form `~/`*PathName*, then *PathName* is taken relative to the user's home directory (as determined by `getenv('HOME', X)` in *SWI-Prolog*).
- If the specifier is of the form `../`*PathName*, then *PathName* is take relative to the parent directory of `DOCDIR`. A prefix of repeated occurrences of `../` is handled analogously.
- Else, the specifier is taken as path name relative to `DOCDIR`.

7.3 Configuration Keys

[*config-key*] `appdata_dir`
: Directory with data files (in Prolog syntax) that are specific to the edition project. Example: `data`.

[*config-key*] `bib_dir`
: Directory with *BibLaTeX* bibliography data files. Example: `bib`.

[*config-key*] `gendata_dir`
: Directory used by the *KBSET* processor for certain generated files. Example: `gendata`.

[*config-key*] `html_output_dir`
: Directory used by the *KBSET* processor for generated HTML documents. Example: `'~/tmp_letters_mini'`.

[*config-key*] `letter_latex_sources`
: List of LaTeX files with letters (`letter` environments [*KLM*]). The suffix `.tex` is omitted. Example: `[letters]`.

[*config-key*] `annotation_latex_sources`
: List of LaTeX files with annotations (`annotation` environments [*KLM*]). The suffix `.tex` is omitted. Example: `[annotations]`.

[*config-key*] `annotation_section_order`
: List of titles of ksections (sections in the `annotation` environment) in the order in which they should appear in final documents. The titles are specified as words separated by single spaces (punctuation and whitespace in the respective matching source document titles is ignored). Example:

```
['Überlieferung',
 'Datierung',
 'Anschrift',
 'Einschluss und mit gleicher Sendung',
 'Vermerke und Zusätze',
 'Varianten',
 'Stellenkommentar']
```

[*config-key*] `factbase_latex_sources`
: List of LaTeX files with fact bases [*KLM*]. The suffix `.tex` is omitted. Example: `[factbase]`.

[*config-key*] `biblatex_sources`
: List of *BibLaTeX* bibliography data files stored in the directory specified as `bib_dir`. The suffix `.bib` is omitted. Example: `[sulzer_sek, sulzer_prim]`.

[*config-key*] `data_sources`
: List of data files in Prolog syntax or in HTML5 stored in the directory specified as `appdata_dir`. The suffix `.pl` is omitted. Example:
`[autographs, images, geostuff, personstuff, 'persontexts.html']`.

  HTML files are useful to represent data values that are maintained as HTML fragments. These files are processed by extracting each element of the form `<div`

`class=xentry-`*`Predicate`* `id=`*`Id`*`>`*`Content`*`</div>` and converting it to a fact

`Predicate``(`*`Id`*`,`*`Content`*`).`

In the fact representation *Id* is converted to the ID used in LaTeX and Prolog (whose special characters my differ from the ID used for HTML, XML and document names), and *Content* is a list of `element`/3 terms in the HTML representation of SWI-Prolog. A predicate *Predicate* can have for a given *Id* at most one such fact.

[*config-key*] `html_options`
List of *key=value* pairs with options for the HTML generation. Example:

```
[maintitle='Beispiel',
 search_action='http://www.mysite.com/cgi-bin/search.cgi/search.html',
 bibsections=[sect(idsek,'Sekundäliteratur',[+sek]),
              sect(idprim,'Primärliteratur',[+prim])]]
```

[*config-key*] `html_extras`
List of files and directories that should be copied to the directory specified as `html_output_dir` in addition to the generated HTML pages. Example:

```
['extras_html/index.html',
 'extras_html/style.css',
 '~/spacerepos/sb_auxfiles/img',
 '~/spacerepos/sb_auxfiles/aut']
```

8 Shell Scripts

8.1 General Remarks

Directory from where to Run these Scripts. All of these scripts are expected to be run from the `DOCDIR` directory.

Determining the *KBSET/Letters* Configuration. The scripts with prefix

`letters_gen_`

are controlled by a *KBSET/Letters* configuration file (Sect. 7), which is determined as follows:

- If an argument is supplied, it is taken.
- Else, if the environment variable `KBSET_LETTERS_CONFIG` is bound, its value is taken.
- Else, the file `kbset_config.pl` in the current working directory is taken.

The script `letters_check_db.sh` proceeds in the same way, except that its optional *second* argument can specify the configuration file.

8.2 Scripts

[*script*] `letters_check_latexdb.sh` *FactBase*.

FactBase is a LaTeX file with just fact base statements [*KLM*]. (The `db....tex` files in the provided example are such files). Outputs some information about syntactic validity and semantic consistency.

[*script*] `letters_gen_annotations.sh`

Generates the file `generated_annotations.tex`, which contains the `annotation` environments in the files configured as `annotation_latex_sources`, ordered in correspondence to the ordering of `letter` environments by `letters_gen_annotations.sh`.

[*script*] `letters_gen_dbs.sh`

Generate Prolog versions of the LaTeX fact bases configured as `factbase_latex_sources`. The generated files are written into the directory configured as `gendata_dir`.

[*script*] `letters_gen_html_bib.sh`

Generates an HTML presentation of the bibliography configured as `bibtex_sources` and an additional data file for use in HTML conversion of letters and annotations. The HTML output is written to the directory configured as `html_output_dir`, the additional file to `gendata_dir`. As a first step, the script invokes `biblatex` in a mode that creates an XML representation of the preprocessed bibliographic data base. This representation is then read-in and further processed by the *KBSET* core system. The result depends on outputs of `letters_gen_html.sh` (to determine the cited works) and vice versa (to determine the bibliography labels used in citations). Thus, to get proper results, `letters_gen_html_bib.sh` should be called between two calls of `letters_gen_html.sh`.

[*script*] `letters_gen_html_extras.sh`

Copies files and directories configured for the `html_extras` key to the directory configured as `html_output_dir`. Typically, these are stylesheets, an index file and images.

[*script*] `letters_gen_html.sh`

Generates a HTML presentation of the project. The output is written to the directory configured as `html_output_dir`. Requires files previously generated by

`letters_gen_reordered.sh`,
`letters_gen_annotations.sh`,
`letters_gen_dbs.sh`, and
`letters_gen_html_bib.sh`.

[*script*] `letters_gen_reordered.sh`

Generates the file `generated_letters.tex`, which contains the `letter` environments in the files configured as `letter_latex_sources`, ordered by date. The ordering takes various forms of incomplete date specifications into account: If the day is unknown, it is placed at the end of the month. If day and month are unknown, it is placed at the end of the year. If the date is specified as *before* or *after* a specific date, it is sorted as if the value for *day* would be one less or one more, respectively, than the specified date. For date ranges, the last value of the range is considered. If letters are considered to have equal dates, they are ordered by lexically comparing the sequence *FromId*, *ToId*, *LetterId* as given in the header of the `letter` environment. (In future version the names of the persons may be considered here instead of the *Id*s).

[*script*] `letters_xindy.sh` *MainFile*

MainFile is a LaTeX main file, written without `.tex` suffix. Invokes the *xindy* indexing system on *MainFile* with suitable options for the indexes created with the *KBSET/Letters* LaTeX styles.

[*script*] `letters_gen_metadata_export.sh`

Generates a file with Prolog facts that represents meta data for the edition project for use in other applications. The meta data are extracted and combined from various sources used for the edition project. This is currently at an experimental stage. In the future, a set of predicates that represent the meta data should be specified. Requires files previously generated as specified for `letters_gen_html`. The output file is `gendata_dir/generated_metadata_export.pl`.

9 Usage Scenarios

9.1 General Notes

Directory from where to Run these Scenarios. All of these scripts are expected to be run from the `DOCDIR` directory.

Assumed File Names We assume two LaTeX main files `main.tex` and `main_devel.tex` as in the example

`KBSET/example/letters_mini`

and explained in Sect. 5. (Of course, in a particular project, respective files can have other names.)

Repeated Runs of `pdflatex`. LaTeX operates with auxiliary files that are created in a processing run and reconsidered in the next run. Thus, LaTeX processors typically have to be invoked several times on the same document. We indicate this in the following usage scenarios by two consecutive runs of `pdflatex`. In practice, during development, often a

single run is sufficient for an acceptable result if there have been only minor document changes since the previous run of `pdflatex`. On the other hand, for production quality results, it is recommended to invoke `pdflatex` even more than two times, say four times to be sure, in these stages.

Why there are so Many Explicit Steps at the Scenarios? It is of course easily possible to combine the listed steps into a single script. However, at each stage there is the possibility of errors or inconsistencies, such that the automated combined processing seems not very useful during the development of editions. Often, during development, the work is only refined with respect to a single stage.

9.2 PDF Generation During Development

1. `$ pdflatex main_devel`
2. `$ biber main_devel`
3. `$ pdflatex main_devel`
4. `$ letters_xindy.sh main_devel`
5. `$ pdflatex main_devel`

9.3 PDF Generation

1. `$ letters_gen_reordered.sh`
2. `$ letters_gen_annotations.sh`
3. `$ pdflatex main`
4. `$ biber main`
5. `$ pdflatex main`
6. `$ pdflatex main`
7. `$ letters_xindy.sh main`
8. `$ pdflatex main`
9. `$ pdflatex main`

9.4 HTML Generation

1. `$ letters_gen_reordered.sh`
2. `$ letters_gen_annotations.sh`
3. `$ letters_gen_dbs.sh`
5. `$ letters_gen_html.sh`
5. `$ letters_gen_html_bib.sh`
6. `$ letters_gen_html.sh`
7. `$ letters_gen_html_extras.sh`

10 Consistency Validation

Consistency checking of an advanced digital scholarly edition is a complex and subtle topic. Here are some suggestions. We assume the LaTeX main file called `main.tex`.

10.1 General Remarks

Basic Consistency Checking with the Standard LaTeX Workflow Of course, the plain LaTeX workflow with `pdflatex`, `biber` and `xindy` already provides some basic validation. Here are some further suggestions and examples.

Output Redirection to a File. Often it is useful to have lengthy output and error log of conversion operations with warnings available as a file. This is possible in the Bash shell with IO-redirection as in the following example:

```
$ letters_gen_html.sh &>tmp_outputs.txt
```

Directory from where to Run these Scenarios. All of these scripts are expected to be run from the `DOCDIR` directory.

10.2 Basic Validation Beyond the LaTeX Standard Workflow

1. `$ letters_check_latexdb.sh factbase.tex`
   `$ letters_check_latexdb.sh dbperson.tex`
   `$ letters_check_latexdb.sh dbwork.tex`
   $ etc., for each LaTeX fact base
2. `$ letters_gen_reordered.sh`
3. `$ letters_gen_annotations.sh`
4. `$ letters_gen_html.sh`

10.3 Validation on the Basis of LaTeX Log Files

First we invoke `pdflatex` twice to create a log file and ensure that cross references are properly handled:

```
$ pdflatex main
$ pdflatex main
```

Now we can search with `grep` for relevant messages in the file `main.log` created by `pdflatex`. For example:

```
$ grep KBSET main.log
$ grep IDX main.log
$ grep IDX main.log | sort -u >tmp_undef.txt
$ grep multiply main.log
$ grep undefined main.log
$ grep undefined main.log | grep Reference
$ grep Warning main.log
$ grep NOTE main.log
```

10.4 Validation of the Bibliography

We call `pdflatex` and `biber`:

```
$ pdflatex main
$ biber main
```

and check the output of `biber` for `ERROR` and `WARNING`. We the call `pdflatex` again:

```
$ pdflatex main
```

because some errors related to the bibliography only show up then. Also the HTML conversion of the bibliography shows possible inconsistencies:

```
$ letters_gen_html_bib.sh
```

10.5 Further Possibilities

Also the XML/HTML-validation of the output of the HTML generation shows inconsistencies, for example duplicate identifiers in a letter. This can be performed, for example, by the program `tidy`, which is also available as free software:

```
$ cd my_html_output_directory
$ tidy -qe letter-l001.html
$ for l in letter*.html; do echo $l; tidy -qe $l; done
```

Further possibilities are dedicated programs to check complex constraints. These could be written in *SWI-Prolog* and take as basis either the Prolog representation of the read-in documents that is used for HTML generation, or the generated HTML, which can be easily read in to term form into *SWI-Prolog*.

11 Using the System from Prolog

11.1 Overview on Relevant Modules

With exception of `letters_xindy.sh`, the scripts described in Sect. 8 invoke *SWI-Prolog*. The predicates underlying the scripts are in the module

`kbset_letters(letters_interface)`.

From there, their functionality can be traced further to the particular modules that realize it. The *KBSET/Letters* core system is loaded into *SWI-Prolog* by loading

`KBSET/core/letters_load.pl`.

If then a *KBSET/Letters* configuration is loaded, for example,

`KBSET/example/letters_mini/kbset_config.pl`,

the system is ready.

Instead of `KBSET/core/letters_load.pl` also `KBSET/core/load.pl` can be used, which loads the same modules as `letters_load.pl` and additional modules for the full functionality of *KBSET*.

Invocation of `extract_letters`/0 effects that a representation of letters, including meta information, by a predicate `letter`/3 and some other predicates exported by the module `kbset_letters(extract_letters)` is created. (Invocation of `extract_letters`/0 requires that intermediate files have been generated before by `letters_gen_annotations.sh`, `letters_gen_letters.sh`, `letters_gen_dbs.sh` or their Prolog equivalents). Meta information on other entities is exported by the module `kbset_letters(data_interface)`. These predicates form a representation of the digital edition as fact base. It is easy to write Prolog programs that realize queries of various forms on such a fact base, or to export portions of it as Prolog fact bases for other applications and further querying.

11.2 Conversion to Further Formats

This internal representation can also be used as basis for conversions into other formats, as shown for HTML. Conversion of identifiers to URIs is implemented in module `kbset_letters(kbmappings)`. Module `kbset_letters(textconvert)` is used for the HTML transformation of the Prolog representation of parsed LaTeX fragments and may be modified for conversions to other formats. So far, there are no specific translations into further formats included with the system.[5] For now, the module

`kbset_letters(export_demos)`

is provided to illustrate how the Prolog fact bases, either present in the system as result of, e.g., `extract_letters`/0, or by loading Prolog files with facts, can be converted to RDF/XML and to certain elements of TEI/XML that represent metadata. In addition, it is sketched there how images with graph representations can be generated from Prolog fact bases with the *dot* tool of *Graphviz*, provided with the module `swilib(dotgraph)`.

[5] In general, specializations of TEI/XML and RDF/XML are relevant there. Actually, it is not yet clear, which particular such specializations formats would be of interest. Criteria would be the integration with other editions and metadata repositories and the use of further tools to generate high-quality presentations.