Take Notes – Take Two

Notes handling module revised and extended

Abstract
Second, revised and extended, version of a module for processing of notes. Notes are classified according to category/subcategory and can contain information about subject, author, date, source, etc. The typesetting of the notes can be filtered according to several criteria. Many aspects of the formatting are easily configurable.

Introduction

Active email lists, as for example the ConTEXt-list, produce many emails discussing some problem or other topics. Often these discussions form threads chaining successive postings, where each posting incorporates much content of the previous one. Collecting all the emails for later reference produces a lot of redundancy. Moreover, the continuous repetition of previous content tends to make reading a tedious business.
Clearly better accessibility can be attained when the threads worth retaining are condensed to a single or a few successive notes, describing the gist of the discussion. This idea motivated the development of this module. It serves as a means to collect, store, select and reproduce the stored notes. Usage of this module is not restricted to storing email discussions, of course. It can be applied to many topics one deems worth remembering.
The above formed the motivation from which the first version of this module originated. The development of software pieces like this is in my case highly governed by the need of the moment. With new (self imposed) tasks at hand, new possibilities literally scream for implementation. This was the case when I turned my attention to researching the origins of my family. The mass of facts to be taken into account (hundreds in the archives of the 18th century alone) could be kept under control only when put into uniform records. Thus an adapted version came into existence as a spinoff from the original module: it could keep track of all persons of interest playing a role in the source documents. Because two nearly identical programs are a bad thing, especially with respect to bug fixing and maintenance, it was decided to combine them in a new and extended `takenotes` module. Also this offered a chance to straighten out the code and revert from a few bad structural decisions.

Overall structure

This module is dependent on other modules: `hvdm-lua`, `hvdm-ctx`, `hvdm-voc` and `hvdm-xml`. They are loaded from within the main module file, now named `hvdm-tak` (from *tak*enotes, as is easily guessed).
Notes are stored in the main document within the root node `<takenotes>` or may be put into separate files to be loaded on the fly. A `<note>` has the following general structure:

```
<note attribute=".."  attribute="..">
  <subject> ... </subject>
  <author> ... </author>
  <source> ... </source>
  <text> ... </text>
  <remark> ... </remark>
  <!-- etcetera -->
</note>
```

This works fine with one `<note>` per file but not with several. In that case the first one is processed, the others are 'forgotten'. In order to have them processed properly they must be be enclosed inside a `<notes>` node which then functions as their root. The same applies to node collections which can be read from a file or buffer, as explained below.

```
<notes>
  <note> ... </note>
  <note> ... </note>
   <!-- etcetera -->
</notes>
```

Thus notes can be typeset from various sources:

```
<takenotes attributes="">
  <!-- Local notes -->
    <notes>
    <note> ... </note> <note> ... </note> ...
    </notes>
  <!-- Included from file -->
    <notes file="somenote.xml"/>
  <!-- Included from directory -->
    <notes folder="somedirectory"/>
  <!-- Included from mtxrun -->
    <notes arg="someparameter"/>
</takenotes>
```

The first three possibilities speak for themselves, but the last one needs an explanation. ConTEXt typesetting is set in motion from the `mtxrun` script. That script has parameters which can be picked up by the executing program. An example will make this clear. Suppose one has a tailored ConTEXt file `template.tex` that should be used to process individual notes, each in their own file. This can be accomplished by running the template file with inside it `<notes file="somenote.xml"/>`, each time changing `somenote.xml` to a new value. For a few hundred files this is a herculean task, to say the least. How much easier if but one command could do them all.[1] The last option above is meant to accomplish just that. In general ConTEXt is called from the command line with:

```
mtxrun --script context file
```

We add our own parameter, naming it for the sake of this example `someparameter`, and run with:

```
--someparameter="somenote.xml" template.tex
```

The template file is then run on `somenote.xml`. In order to do this for a great many of files, all what is needed is a shell script that repeatedly executes the `mtxrun` with a shell parameter for `file`. At the end of this article two `bash` shell scripts are outlined that together accomplish this task.

Note structure
Minimal template for a note:

```
<note
  category="" subcategory="" tag="" date=""
  key="" language=""  text="" obsolete="">
  <!-- child nodes -->
</note>
```

The data pertaining to a note are split between the attributes and its child nodes. Attributes are used to classify the note and will serve as keys in the filtering of notes during the processing stage. These attributes have the following meaning.

- ⋆ `category` – *Mandatory* Freely chosen designation for the topic or type of document under which the subsequent notes are categorized by the user.
- ⋆ `subcategory` – A category can be refined into subcategories, making finer meshed searches possible. An example from my collection of genealogical data: `category="register"`, `subcategory="marriage"`.
- ⋆ `tag` – Optionally notes can be tagged in the (left) margin with this tag, usually a single short word.
- ⋆ `date` – The date pertaining to the note. Usually the date on which the content was produced originally or the date of the event it describes.
  - A date of 6 or 8 digits will be interpreted as `yyyymmdd` or `yymmdd` respectively; the latter designating a date in the current 21st century. Thus `17530427` will be interpreted as the 27th of April 1753 and `190401` as the 1st of April 2019.
  - Other acceptable date formats are `dd-mm-yyyy`, `d-m-yyyy`, `yyyy-mm-dd`.
  - Four consecutive digits will unsurprisingly be treated as a year.
  - Two years separated by one or more dashes, possibly surrounded by whitespace, constitute a range of years; sorting is on the first year of the pair. An example is "1750 - 1780".
  - Dates given as '`May 12,1981`' are printed as such, but cannot take part in date sorting.
  - A missing date will generate a warning, but it can be appropriate to leave out the date. In that case use `date="nodate"` to silently suppress output of a date.
  - Dates found in error will generate a warning.
- ⋆ `key` – A number of space and/or comma separated keywords that can be used to filter out notes of particular interest.
- ⋆ `language` – Signifies in the standard two-letter code the language in which the content of the note is written. Needed only when that language differs from the language in which the full document is typeset and should be reflected in the output. The internationalization of certain terms in the output uses this value.
- ⋆ `text` – Even if the user has chosen to generally suppress `<text>` nodes (see below), still some notes will be meaningless without it. Setting `<note text="on">` (or `yes`, `true`) forces the `text` nodes out for this specific note. Conversely, the text on the note can be suppressed regardless of the selected setting with `text="off"` (or `no`, `false`).
- ⋆ `obsolete` – Flags obsolesce of the note and is used to suppress its output without the need to remove the note physically. It then can be easily reinstated when need arises. Values `on`, `yes`, `true` and `off`, `no`, `false` makes the note obsolete/active, the latter being the default. For obsolete notes, when their output is selected, a bare minimum of child nodes is displayed.

Child nodes and their presentation
The child nodes carry the information proper. A few standard child nodes are implemented already, chosen as being the most general. Others can be added by the user at will. We shall refer to the child nodes of `<note>` as 'legends'.
When typeset, the legends precede the text content contained in `<text>` nodes. They adapt their heading to the language set for the ConTEXt document or the `language` attribute on the `<note>`. Provided, of course,

the language equivalents are known to the module (as is the case for english, dutch, french and german for the standard legends). It is not difficult to add other translations in other languages (see the explanation of using <vocabulary> later on).
Legends <subject> and <abstract> will be placed but once in the output and then their first occurrence is taken, the others being ignored. Legends <alert> and <source> on the contrary will be gathered from everywhere in the note and grouped together. The other legends are not constrained.
Typesetting is done in a certain order. First comes the information taken from the note's attributes. Then the legends in the order determined by the program. Note however, that the order in which they are put inside the <note> can be chosen freely. The order in the enumeration below roughly reflects the ordering in the output.

- ⋆ <subject> – *Mandatory* The very first legend containing a short description or title of the subject of the note; only its first occurrence is used. The subjects of all notes are optionally combined in a list from where they can be reached by clicking on them. The subject is never suppressed, not even on obsolete notes when these are printed.
- ⋆ <author> – Author(s) of the information and/or the writer(s) of the note, presented together comma separated.
- ⋆ <source ref="" link=""> <url> – Carries the source or location of the information. Generally containing a link to an URL, either local or on the internet. It can be a posting on the internet or a reference into a (public) archive.
  Two attributes are provided for locating the source: `ref` is what will be printed and *must be present*, `link` is the underlying URL. The link attribute (if not absent or an empty string) is checked and an error message presented if it cannot be reached. Otherwise an active link is made so that clicking on it brings the underlying document in view. *Beware: spaces in these links must be explicitly coded as %20 or else the link will not be found!*
  Sometimes it is preferred to embed the <source> references within the text rather then to group it at the <note> level. Then the reference is printed both in the <source> group as well as locally. The first occurrence will carry the interactive link, the latter being bare plaintext. Also, in that case the content of the node is not ignored as is otherwise done; this content appears behind the legend en is useful for example to annotate the source.
  The <url> is synonym to <source> and has the same effect. Differentiation between them allows one to activate/deactivate them separately.
- ⋆ <person omit=""> – This is a legend with child nodes, meant for the description of persons and derived from my genealogical interest.
  The legend has one attribute `omit`. Setting this to `on` (or `yes`, `true`) marks this person eligible to be omitted if chosen so on a takenotes run. It allows one to register all persons figuring in a document and at the same time differentiate between the more important ones and others. A minimum of child nodes is provided, but others can be added by including their node names in the <takenotes> attributes (see below). Be aware of the fact that nodes not marked as such will be ignored inside the <person> node; otherwise there would be too much interference with the formatting.
  - `name` – The name of the person. It is an option to collect these names in a list that then will be found at the end of the <takenotes> run behind the list of subjects.
  - `title`, `surname`, `between`, `name` – Alternative to putting the full name inside <name> is to split it up into [<title>] <name> [<between>] <surname> (`title` and `between` being optional). The presence of a not empty <surname> triggers this alternative.
  - `named` – Especially in the past, people were often not very much concerned with spelling issues. Therefore names in documents will differ in spelling or are clearly misspelled. Also, in certain documents a latinized form of the name is used; for example in church archives recording baptisms one may find 'Adrianae' meaning 'from Adriana'. Another variant is 'Joannes' for 'Johannes', not to speak of abbreviations like 'Jo:es'. To cope with all these variations `named` is provided. It is useful for registering the name exactly as encountered in the document, while at the same time keeping the standard name in `name`. This entry will be typeset within parentheses behind the proper name.
  - `relation` – People are related to each other, such as 'son of', 'spouse of', etc. Adding their relation helps to differentiate between people with the same name but different ancestry.
  - `role` – The way people act in the document, for example as buyer of a house, bride in a marriage, deceased in a burial.
  - Add new ones through the `on` attribute of <takenotes>, because child nodes not being registered as legend will be ignored.[2] Optionally add their translation to the vocabulary (see below).
- ⋆ <user-legends-nodes> – The number of legends preprogrammed is limited and more often than not insufficient for the task at hand. The module has to be informed then in order to be able to place the

additional legends correctly. The method chosen is a very simple one. To add a <location> legend for example, list that node's name in the on attribute: <takenotes on="location">. Thereafter <location> is recognized as a legend and typeset as such.
On the other hand, nodes present but unwanted must be marked as such or their content will appear unstructured in the legend list. To avoid this they must be enumerated in the off attribute: off="location" will suppress the <location> node. The alert and subject have been made exempt of this mechanism, guaranteeing they will not be excluded.
Because the underlying definitions are made globally, there is no need to repeat them within the same ConTEXt run.

- <non-legend-nodes> – Comprises all XML-nodes not programmed in this module and not mentioned in either the on or the off attributes. Nothing special is done.
- <group> – Sometimes one may feel the need to meddle with the strict order and selection of nodes at the note level. Then enclose the content with a <group> (and pray all may proceed as intended).
- <alert legend=""> – This is a remark typeset in the alertcolor (see below). Where remarks can be optionally suppressed alerts will be suppressed in obsolete notes only, but otherwise they are always present.
- <remark legend=""> – Meant for placing remarks or other additional information not having a place in the regular text portion of the note. Attribute legend takes the place of the heading 'remark' in the output. An example is legend="todo" for a reminder.
- <abstract> – Speaks for itself, an abstract of the contents of this note. The first one is printed, later occurrences are ignored.
- <text title="" indent=""> – Textual content of the note, in one or more <text>-blocks, all printed in the order of occurrence.
  When title is present, it is typeset centered at the start of the text block. The indent attribute can be used to explicitly switch indenting on or off locally.

Takenotes attributes

The root node <takenotes> effects the extraction and presentation of the collection of notes.

Selection of notes

The number of notes can grow so large that finding something specific becomes tedious, especially in a printed document. Here two strategies can be followed. The first is examining the resultant pdf in a program as Adobe Reader and using its search facilities. The second is to narrow the output to just the notes of interest. In order to accomplish that, a variety of selection criteria can be applied, either separately or in combination.

- The first set of selection criteria is based on the value of attributes on the enclosing <note attributes>. Selections can be made on category, subcategory, a range of dates, key and obsolescence.
- The second type of selection looks into the content of the note. A search pattern is applied to the full content and the note is displayed when that pattern matches somewhere. Both case sensitive and case insensitive searches can be initiated. Because the searching is executed in Lua, not just fixed patterns but also more complicated general patterns are possible.

Most of the custom settings are made globally in order to facilitate their setup in subsequent <takenotes>'s in one ConTEXt run; for example each in its own chapter. The selection criteria are an exception to this.
Attributes steer the selection of notes according to criteria related to the attributes on the note. The selection criteria work independently from each other and together let pass those notes that conform to all of them.

- category, subcategory – When present and not empty the value of these attributes takes part in a selection based upon the category/subcategory attribute on each <note>; the comparison is case insensitive.
- from and until – Attribute from carries the lowest date on the <note> that will be selected. Conversely until selects the last date included in the output. If both are present they constitute an interval. Dates must be formatted according to the rules explained above. Free format dates or notes without a date will be assigned the lowest date the module provides (1-1-4712 BC).[3]
- key – Will be matched to the comma or whitespace separated list of keywords on the <note>. If there is match, the note will be selected. This matching is case sensitive.
- select – The value of this attribute is a Lua search pattern applied to all sub-nodes within the <note>. A matching pattern contributes to typesetting of the note, in combination with the other selection criteria that are activated.
  When there are uppercase letters in the search pattern a case sensitive search is done, otherwise it is case insensitive.[4] If <text> blocks are suppressed they are left outside the search and therefore do not influence the selection.

Selection of features

Typesetting of the notes is determined twofold. First one may choose which legends will appear and which

not. This regards both the legends builtin as well as those provided by the user, as already has been outlined above.
Second are flags that influence the output in many respects. Activation and deactivation are done with the `on=""` and `off=""` attribute respectively. Values within the attribute must be separated by commas, may have spaces between them, but not divided with a newline.
Flags are capitalized, legend names all lowercase. A catalogue of the possibilities follows.

Choosing the legends

- By default on are the legends:
  `abstract`
  `alert` (always)
  `author`
  `person`, `relation`, `role`
  `remark`
  `source`
  `subject` (always)
  `text`
  `url`
- By default on are the flags:
  `Category`
  `Date`
  `Omit`
  `Sort`
  `Sortup`
- By default off are the flags:
  `Filename`
  `Key`
  `Namelist`
  `Subjectlist`
  `Newpage`
  `Nobreak`
  `Number`
  `Obsolete`
  `Tag`
  `Verbose`
- `Namelist` – If `<person>` nodes are present, their `name` will be collected and afterwards put in a two-column list in alphabetic order. When this list is present a link to it will appear at the top of each page.
- `Subjectlist` – The subjects are collected and output in alphabetical order. If the `Number` attribute has been set, the subjects in the list will be tagged with the sequence number of the corresponding note. If `Date` is selected, the note dates are attached.
  When the list of subjects is present a link to it will appear at the top of each page. Furthermore, the lines in the subject list are active and carry to the page in the document where this specific subject resides. All links are local to the current instance of `<takenotes>` as is the case for the list of names.

Tailoring the appearance

- `Newpage` – Start a new page for each note.
- `Nobreak` – Prevents breakup of a note over a page boundary by placing it in a framed box. Be aware of the fact that long notes will play havoc with page boundaries.
- `Filename` – Turns on a note separator with the name of the file in it. It can be combined with `Number`.
- `Number` – Turns on a separator with the sequence number of the current note in it. Can be combined with `Filename`.
- `Obsolete` – Include obsolete notes in the output which are otherwise ignored. If included their content is curtailed to the subject and the text color altered (by default dimmed to gray).
- `Omit` – Its use is discussed with the `<person>` child node.
- `Sort`, `Sortup` – Notes are sorted on their date in ascending order by default. Turn that into descending order by adding `Sortup` to the `off` attribute. Notes with a missing or empty date are grouped together.

Style and color

The content of the notes can be given different font and style from the rest of the document. An example of the use of a different font for the text would be to typeset program code in a monospaced font.
The following values are available for style options: `bf`, `it`, `bi`, `bs`, `sl`, `tt`, `sc`, `rm`, `ss`, `tf`, `tfa`, `tfb`, `tfc`, `tfd`, `tfx`, `tfxx`, `ttx`, `ttxx`, `os`, `hw`, `cg`, `bold`, `italic`, `bolditalic`, `italicbold`, `slant`, `slanted`, `boldslanted`, `slantedbold`, `smallcaps`, `oldstyle`, `mediaeval`, `normal`, `serif`, `regular`, `roman`, `sans`, `sansserif`, `mono`, `type`, `teletype`, `handwritten`, `calligraphic`, `big`, `verybig`, `heavy`, `veryheavy`, `small`, `tiny`, `smallmono`, `tinymono`.
These values translate to corresponding font commands. Several font commands may be combined as for example `legendstyle="bf,os"`, but not all combinations are effective. Be aware of the fact that not all of the above can be considered pure style options. For example the use of `tt` might not change the font family but has a profound influence on the look of the text nonetheless.

- `abstractstyle` – Used to set a different style for the text in the abstract.
- `legendstyle` – Style used for the legends.
- `textfont`, `textstyle` – Font and style for the text blocks, abstracts inherit the same font but may differ in style.

Colors may be left at their default values, switched to the (black) text font by providing the value `none` or given a color at will.

- `alertcolor` – Color used for the alerts.
- `backgroundcolor` – When set the background of the notes will be colored.

- ⋆ `linkcolor` – Color used for active links.
- ⋆ `obsoletecolor` – Color used for the obsolete notes when printed.
- ⋆ `omitcolor` – Color used for the `<person>` node when marked for omission in case they are printed.

Languages
Elements of the output can be internationalized through definition and use of one or more vocabularies. This applies to the language in which the legends are typeset, but depending on the effort one wishes to invest many parts of the text can undergo internationalization. The component effectuating this is the module `hvdm-voc` which is incorporated by the module `hvdm-xml`.
It all starts with the creation and filling of a vocabulary. By default there is a vocabulary `languages` defined (in part) by the code below. A named vocabulary is automatically created the first time one attempts to add something to it.

```
<vocabulary name="myvocab">
<word add="dutch">
  <en>dutch</en>
  <nl>nederlands</nl>
  <de>niederländisch</de>
  <fr>n&eacute;erlandais</fr>
</word>
</vocabulary>
```

In the document the code shown above is loaded from a buffer or a file with:

```
<vocabulary name="myvocab" buffer="aBuffer"/>
<vocabulary name="myvocab" file="aFile"/>
```

Individual translations may be loaded one by one with:

```
<vocabulary name="myvocab" add="greek">
  <en>greek</en>
  <nl>grieks</nl>
  <de>griechisch</de>
  <fr>grec</fr>
</vocabulary>
```

Translations are retrieved by:

```
<vocabulary name="myvocab" get="greek"/>
<vocabulary name="myvocab" Get="greek"/>
<vocabulary name="myvocab" GET="greek"/>
```

With the current language being ‘en’, it results in greek, Greek and GREEK. However, after changing to german the results change accordingly:

```
<vocabulary use="german"/>
```

Now it is griechisch, Griechisch and GRIECHISCH. Similarly the default vocabulary to use can be set:

```
<vocabulary set="myvocab"/>
```

If the document is changing vocabularies often, it might be safer to use the vocabulary’s name on each node explicitly. The more because a word not known in the vocabulary at hand, will appear untranslated. Most of the operations on vocabularies can be called directly from a ConTEXt-document with macro’s `translate`, `Translate`, `TRANSLATE`, etc. See the code of the module for the details.

Example
Two examples of a note. The first is in the Dutch language as designated by the attribute `nl`. Note by comparing the XML and its output, that flagging a note as obsolete suppresses everything except the subject title when printed nonetheless. The second is an example from a set of genealogical data. It also serves the purpose of showing how HTML-elements from module `hvdm-xml` can be used.

onderwerp: Cryptografie cursus
datum: 1-1-2018
categorie: Cryptografie/cursus
auteur: dr. Hans van der Meer
opmerking: This course is in Dutch.
samenvatting: Cursus cryptografie
Algemene inleiding tot de cryptografie en cryptoanalyse met oefeningen voor studenten.

Note the difference in output between an active note and an obsolete one.

Vervallen
onderwerp: Cryptografie cursus

```
<?xml version="1.0" encoding="UTF-8"?>
<takenotes on="Nobreak"
  backgroundcolor="lightgray">
<note date="20180101"
  category="Cryptografie" subcategory="cursus"
  key="cryptografie cryptoanalyse"
  language="nl" obsolete="no">
  <subject>Cryptografie cursus</subject>
  <author>dr. Hans van der Meer</author>
  <abstract>Cursus cryptografie</abstract>
  <text>
    Algemene inleiding tot de cryptografie en
    cryptoanalyse met oefeningen voor studenten.
  </text>
  <remark>This course is in Dutch.</remark>
</note>
</takenotes>
```

subject: Marriage of Arnoldus Vorst and Maria Catharina van Hoven
date: 11-1-1727
category: archive/marriage
source: HGA 0321-01 inv.16 f.189
location: The Hague
▷ *person:* Arnoldus Vorst
relation: father of Maria Vorst
role: groom
age: 24
▷ *person:* Maria Catharina van Hoven (Hove)
relation: mother of Maria Vorst
role: bride
age: 22
remark: Married pro deo at town hall.

```
<?xml version="1.0" encoding="UTF-8"?>
<takenotes on="Nobreak, location, age"
  backgroundcolor="lavenderblush"
  from="1-1-1700" until="31-12-1750">
<color name="lavenderblush"
  green=".941176" blue=".960784"/>
<note
  category="archive" subcategory="marriage"
  date="11-1-1727" key="vorst" language="en">
<subject>
  Marriage of Arnoldus Vorst and
  Maria Catharina van Hoven
</subject>
<location>The Hague</location>
<source ref="HGA 0321-01 inv.16 f.189"/>
<ul sym="4"><li>
<person>
  <name>Arnoldus Vorst</name>
  <named></named>
  <relation>father of Maria Vorst</relation>
  <role>groom</role>
  <age>24</age>
  <profession>carpenter</profession>
</person>
</li><li>
<person>
  <name>Maria Catharina van Hoven</name>
  <named>Hove</named>
  <relation>mother of Maria Vorst</relation>
  <role>bride</role>
  <age>22</age>
</person>
</li></ul>
<remark>Married pro deo at town hall.</remark>
</note>
</takenotes>
```

Note here that the location and the age have been activated with `on="location, age"` and thus placed on each person, while the undeclared profession is absent.

Bash scripts for using templates

To get an impression of the scripts that run a template with the `arg` option on several <note> containing files, an extract follows below. The full scripts can be obtained from the internet. One such place is the *publications page* on `hvandermeer.com` (use the link *ConTeXt module distribution*).

```
#!/bin/bash
#   ¢Hans van der Meer hvandermeer.com
#   Run template tex-file on targets through notesone script.
for filename in "$@"; do
    notesone "$TEMPLATEFILE" "$filename"
done
```

```
#!/bin/bash
#   ¢ Hans van der Meer hvandermeer.com
#   Process one file with ConTeXt through template.
mtxrun --script
  context --$TARGETARGUMENTNAME="$2" "$1" >/dev/null
#   Change the output pdf to the target basename.
mv "$SOURCENAME.pdf" "`dirname "$2"`/$TARGETNAME.pdf"
```

Notes

1. For the fans: Yes, you can hear The Lord of the Rings here.
2. This may seem overly restricted, but it prevents the module of becoming too complicated and is at the same time beneficial for a clear presentation of these data.
3. For those who wonder: the lowest value for the Julian date.
4. In Lua patterns elements such as %A, %S are used to suppress certain categories of characters and thus are distinct from uppercase text. The search mechanism implemented is aware of that distinction.

Hans van der Meer
havdmeer@ziggo.nl