Translations from a Vocabulary

Handling translation into various languages

Abstract
Formerly part of the module hvdm-xml but now split off into an independent module with its own description. Used for making other modules language sensitive. The module is especially tailored for XML use.

Introduction

Elements of the output can be internationalized through definition and use of one or more vocabularies. This module allows for a flexible and adaptable translation of individual words. The component effectuating this is the module `hvdm-voc`. Its interface is meant to be accessed from XML as well as through ConTEXt-macros.
This module certainly is *not* a full blown translator. Its scope is restricted to the translation of individual words, not even plural forms are automatic and must be added separately. But although simplistic in nature, it provides for the automatic adaptation of certain keywords to a change of language.

It all started with the construction of a database in XML format with historical facts about my ancestors. Each fact resides in a separate file to be processed by my note processing module `hvdm-tak`. Everything is enclosed in an XML node with suitable node names chosen. Since the lingua franca in computing is the English language, it seemed natural to use this for these names. Thus each person in my notes became to be described by a `<person>` node, for instance:

```
<note>
    <person>
        <name>Arnoldus van der Meer</name>
        <age>28</age>
        <profession>seller of mineral water</profession>
    </person>
    <!– other nodes –>
</note>
```

Typeset in ConTEXt with `\language[en]` such a note looks as in the next figure. Note the english keywords at the left edge corresponding to the node names. The remainder of the text is in Dutch, as might be expected for material taken from five centuries of dutch archives.

note-1 — file: events/hga-0402-422-1-99.xml

subject: Schuldinning door Arnoldus van der Meer
date: 5 November 1779
category: event/weeskamer
key: vandermeer blauwsonnevelt prins
author: Hans van der Meer
source: Haags Gemeentearchief 0402-01 inv.422 pdf-1 p
location: Den Haag
person: Arnoldus van der Meer
role: schuldinner
person: Francina Prins
relation: weduwe van Frederik van Blauwsonnevelt
role: erfgename
person: Frederik van Blauwsonnevelt (Fredrik van Blaau
relation: overleden van echtgenoot Francina Prins
abstract:
Arnoldus van der Meer neemt op zich om voor de Blauwsonnevelt en hun twee minderjarige kindere van f325–15 te innen.

Language set to English.

Not all members in my family did like the english words interspersed between the text in Dutch. Thus arose the idea for this translator. The keywords with their translation were put in a vocabulary as described below. In the notes module their typesetting is enclosed in a `\translate` macro call. The result is shown in the second figure, again note the keywords at the left edge.

notitie-1 — file: events/hga-0402-422-1-99.xml

onderwerp: Schuldinning door Arnoldus van der Meer
datum: 5 november 1779
categorie: event/weeskamer
sleutel: vandermeer blauwsonnevelt prins
auteur: Hans van der Meer
bron: Haags Gemeentearchief 0402-01 inv.422 pdf-1 p.
plaats: Den Haag
persoon: Arnoldus van der Meer
rol: schuldinner
persoon: Francina Prins
relatie: weduwe van Frederik van Blauwsonnevelt
rol: erfgename
persoon: Frederik van Blauwsonnevelt (Fredrik van Blaa
relatie: overleden van echtgenoot Francina Prins
samenvatting:
Arnoldus van der Meer neemt op zich om voor de Blauwsonnevelt en hun twee minderjarige kindere van f325–15 te innen.

Language set to Dutch.

But (inevitably) I became a bit sloppy. Dutch words crept in as node names for new properties which were added in the course of the genealogical investigations:

```
<note>
    <person>
        <name>...</name>
    </person>
    <samenvatting>an abstract</samenvatting>
</note>
```

The following figure shows what happens if the note is typeset in the english language again. That one keyword *samenvatting* strangely deviates from the others.

person: Frederik van Blauwsonnevelt (Fredrik van Blaau
relation: overleden van echtgenoot Francina Prins
samenvatting: Arnoldus van der Meer neemt op zich om
erik van Blauwsonnevelt en hun twee minderjarige kind
taal van f325-15 te innen.

Language set to English with Dutch node.

Although changing node names to their english equivalents is a simple action in an editor, it provided the incentive to make the translator module a bit more general. Instead of translating from one language only, it should allow translation between any two languages in the vocabulary. Look at the fourth figure where this flexibility is demonstrated by changing to `\language[de]`.

—— *Notiz-1 — Datei: events/hga-0402-422-1-99.xml* ——
Thema: Schuldinning door Arnoldus van der Meer
Datum: 5 November 1779
Kategorie: event/weeskamer
Schlüssel: vandermeer blauwsonnevelt prins
Autor: Hans van der Meer
Quelle: Haags Gemeentearchief 0402-01 inv.422 pdf-1 p
Ort: Den Haag
Person: Arnoldus van der Meer
Rolle: schuldinner
Person: Francina Prins
Beziehung: weduwe van Frederik van Blauwsonnevel
Rolle: erfgename
Person: Frederik van Blauwsonnevelt (Fredrik van Blaau
Beziehung: overleden van echtgenoot Francina Prins
Zusammenfassung:
Arnoldus van der Meer neemt op zich om voor de
Blauwsonnevelt en hun twee minderjarige kindere
van f325-15 te innen.

Language set to German.

XML Interface

Everything starts with the creation and filling of a vocabulary. By default the module provides a vocabulary to which one can add translations, but it is possible to create others as has been done in the code below. A new vocabulary is created the first time its name appears on a `<vocabulary name="name">` node, further such calls with that name are silently ignored. Creation of a vocabulary will *not* make it automatically the current vocabulary.

That should be done by the separate `set` attribute as in the example below.

```
<vocabulary name="myvocab" set="myvocab">
    <word>
        <en>dutch</en>
        <nl>nederlands</nl>
        <de>niederländisch</de>
        <fr>néerlandais</fr>
    </word>
</vocabulary>
```

Load the above data from a buffer or a file with:

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

Vocabularies are switched with the `set` attribute. A value 'default' for the name performs a switch back to the default vocabulary installed by the module.[1] The code below illustrates how to set and retrieve the names of the current vocabulary and language.

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

The results are vocabulary = *myvocab* and language = *en*. Note that attributes `name` and `set` behave differently. When the named vocabulary does not yet exists the former will create a vocabulary with that name, whereas the latter will issue an error message instead. When both attributes are present the vocabulary is created first and then made current.

Individual translations can be added to any named vocabulary, but when there is no `name` attribute on `<vocabulary>` the current vocabulary will be extended.

```
<vocabulary>
    <word>
        <en>greek</en>
        <nl>grieks</nl>
        <de>griechisch</de>
        <fr>grec</fr>
    </word>
</vocabulary>
```

With the current language being *en* this results in greek, Greek and GREEK. Changing the language setting to german with

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

will change to griechisch, Griechisch and GRIECHISCH.

Translations are retrieved by a `<vocabulary>` node with attributes `get`, `Get` and `GET`. The three variants select the corresponding case variants. Presence of a `use` attribute translates into that language but leaves the current lan-

guage setting unchanged. For instance:

```
<vocabulary get="greek"/>
<vocabulary use="nl" Get="dutch"/>
<vocabulary use="fr" GET="english"/>
```

produce griechisch, Nederlands and ANGLAIS. The presence of a 'get'-ter forces the change from attribute `use` to be locally confined. Although the last language accessed here was from `use="fr"`, the current language *de* has not changed.

The vocabulary is set up in such a way that translations between all language pairs are possible. In itself that sounds nice, but what if a synonym has to be added? For example, besides 'dutch' translated into 'niederländisch', we want the twoletter code 'nl' to be translated into 'niederländisch' too.
The problem here is the following. Addition of 'nl' in the same manner as demonstrated above, will overwrite cross translations already present instead of merely adding the equivalents for 'nl'. The solution is simple. Use `<word add="nl">` and the enclosed translations will be taken for synonyms. In this manner 'nl' is added to the vocabulary without generating cross translations. We will find for instance 'nl' translating into *niederländisch* just as happens when translating 'dutch' to german.

```
<word add="nl">
    <en>dutch</en>
    <nl>nederlands</nl>
    <de>niederländisch</de>
    <fr>néerlandais</fr>
</word>
```

A problem still remains with this translation scheme. Let us add translations for icelandic, switch to dutch and see what `get` and `Get` translations do: *ijslands* and *Ijslands*. The latter is wrong because in dutch the 'ij' counts for one letter! Thus 'Ijslands' should have been 'IJslands'. Luckily the solution is not problematic. Add an extra node for ijsland → IJsland as a synonym with an adapted language code `Nl` The first letter of `nl` now being in uppercase. Instead of raising the first letter of the translation only, the translator then uses the alternative definition:

```
<Nl>IJslands</Nl>
```

With this addition to the vocabulary we now get *IJslands* as it should be. The same problem arises for letters such as the ç in français leading to FRANçAIS instead of FRANÇAIS or the ä in Niederländisch. Here we could have solved it with other exceptions as `<FR>FRANÇAIS</FR>`, but instead the upper-lowercase translator has been made a little bit smarter. It knows how to do a case change for letters like é, à, ü, ç.

ConTEXt Interface

Although primarily developed for use in an XML environment, it boils down to calling into TEX code. It is therefore always possible to fall back onto the underlying macros. The following are the API calls available.

- `\VocabularyCreate[#1]` – creates a vocabulary named in `#1` if it does not yet exists, otherwise do nothing. Note that the current vocabulary is not changed, that should be done explicitly with the set macro.
- `\VocabularyDelete[#1]` – use this in the rare case one wishes to get rid of a vocabulary. The default and the current vocabulary cannot be deleted. Nor is it possible to remove items from a vocabulary once they have been added.
- `\VocabularySet[#1]` – the vocabulary named `#1` will be made the current one. Reset to the module's default vocabulary by calling with an empty parameter.
- `\Vocabulary` – the name of the current vocabulary. Example: the current vocabulary is *myvocab*.
- `\VocabularySetLanguage[#1]` – the two-letter language code makes it the current vocabulary language.[2] An empty argument will set it to the value of `\currentlanguage`.
- `\VocabularyLanguage` – retrieves the two-letter language code of the current language. Example: after changing to french by calling `\VocabularySetLanguage[fr]` the current language at this point is *fr*.
- `\VocabularyLoadFromBuffer[#1]` `\VocabularyLoadFromString[#1]` `\VocabularyLoadFromFile[#1]` – these macros load data as XML nodes from string, buffer and file. Example: prepare with `\startbuffer[spanish]` a buffer to add spanish:

```
<vocabulary>
    <word>
        <en>spanish</en>
        <nl>spaans</nl>
        <de>spanisch</de>
        <fr>espagnol</fr>
    </word>
</vocabulary>
```

and load it with `\VocabularyLoadFromBuffer[spanish]`. Now translation of 'spanish' in *fr* will be *espagnol*. Similarly use `\VocabularyLoadFromFile[italian.xml]` from the prepared file `italian.xml` and obtain *italien* for 'italian'.

▷ `\translate{#1}`
`\Translate{#1}`
`\TRANSLATE{#1}` – translate their argument into the current language with no case change, first letter uppercase, all letters uppercase, respectively. Example: `\TRANSLATE{dutch}` in the current language *fr* results in *NÉERLANDAIS*. But an absent translation returns its argument unchanged as in `\Translate{japanese}` is *Japanese*.

▷ `\VocabularySetLanguageDefault[#1]`
`\VocabularyLanguageDefault`
`\translateDefault[#1]` – There are situations where one language is special. An example is found in the module mentioned in the introduction. Nodes `<author>`, `<person>`, etc. need special treatment in the program. This is accomplished by attaching a flag. Without a common default language it would have been necessary to flag all occcurrences of `<author>`, `<auteur>`, `<Autor>`, etc. separately. By using `\translateDefault` this can be avoided because it enables the programmer to collect all occurrences into a common language. An empty argument `\VocabularySetLanguageDefault[]` sets to the value of `\currentlanguage`. Note that setting of the default language is done globally.

Availability

The module and its supporting modules can be downloaded from my site `hvandermeer.com/publications.html`. The TEX-stuff resides in section "Articles on TeX", downloads are a little below in the link "ConTeXt module distribution".

Notes

1. Note that a vocabulary named 'default' cannot be used and will raise an error if one tries to do so.
2. Do not be tempted to try 'Fr' or 'FR' because the module will silently convert both to 'fr'.

Hans van der Meer
`havdmeer@ziggo.nl`