`smultiling.sty`: Multilinguality Support for STeX

Michael Kohlhase
Jacobs University, Bremen
`http://kwarc.info/kohlhase`

April 22, 2014

Abstract

The `smultiling` package is part of the STeX collection, a version of TeX/LaTeX that allows to markup TeX/LaTeX documents semantically without leaving the document format, essentially turning TeX/LaTeX into a document format for mathematical knowledge management (MKM).

The `smultiling` package adds multilinguality support for STeX.

Contents

1   Introduction

The `smultiling` package adds multilinguiality support for SТEX, it is essentially a wrapper around the `babel` package but allows specification of languages by their ISO 639 language codes.

2   The User Interface

The `smultiling` package accepts all options of the `babel.sty` and just passes them on to it. The options specify which languages can be used in the SТEX language bindings.

3 Implementation

3.1 Class Options

To initialize the `smultiling` class, we pass on all options to `babel.cls` and record which languages are loaded by defining `\smul@⟨language⟩@loaded` macros.[1]

langfiles

The `langfiles` option specifies that for a module ⟨mod⟩, the module signature file has the name ⟨mod⟩`.tex` and the language bindings of language with the ISO

639 language specifier ⟨lang⟩ have the file name ⟨mod⟩`.`⟨lang⟩`.tex`.[2]

```
 1 ⟨*sty⟩
 2 \newif\if@langfiles\@langfilesfalse
 3 \DeclareOption{langfiles}{\@langfilestrue}
 4 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{babel}
 5 \@namedef{smul@\CurrentOption @loaded}{yes}}
 6 \ProcessOptions
 7 ⟨/sty⟩
 8 ⟨*ltxml⟩
 9 # -*- CPERL -*-
10 package LaTeXML::Package::Pool;
11 use strict;
12 use LaTeXML::Package;
13 DeclareOption('langfiles',sub {AssignValue('smultiling_langfiles',1,'global');});
14 DeclareOption(undef,sub {PassOptions('babel','sty',ToString(Digest(T_CS('\CurrentOption')))); }
15 ProcessOptions();
16 ⟨/ltxml⟩
```

We load `babel.sty`

```
17 ⟨*sty⟩
18 \RequirePackage{etoolbox}
19 \RequirePackage{babel}
20 ⟨/sty⟩
21 ⟨*ltxml⟩
22 RequirePackage('babel');
23 ⟨/ltxml⟩
```

3.2 Handling Languages

\smg@select@language

This macro selects one of the registered languages by its langauage code by setting the internal `\smg@lang` macro to the argument and then runs the actual selection code in `\smg@select@lang`. This internal code register is only initialized there, the code is generated by the `\smg@register@language` macro below.

```
24 ⟨ltxml⟩RawTeX('
25 ⟨*sty | ltxml⟩
26 \newcommand\smg@select@lang{}
27 \newcommand\smg@select@language[1]{\def\smg@lang{#1}\smg@select@lang}
```

---

[1] EDNOTE: @DG: We also want to do that in LaTeXML
[2] EDNOTE: implement other schemes, e.g. the onefile scheme.

\smg@register@language   \smg@register@language{⟨*lang*⟩}{⟨*babel*⟩} registers the `babel` language name ⟨*babel*⟩ with its ISO 639 languge code ⟨*lang*⟩ by extending the \smg@select@language macro.

```
28 \newcommand\smg@register@language[2]%
29 {\@ifundefined{smul@#1@loaded}{}{\appto\smg@select@lang%
30 {\expandafter\ifstrequal\expandafter\smg@lang{#1}{\selectlanguage{#2}}{}}}}
```

Now we register a couple of languages for which we have `babel` support. Maybe we have to extend this list with others. But then we have to extend the mechanisms.

```
31 \smg@register@language{af}{afrikaans}
32 \smg@register@language{de}{ngerman}
33 \smg@register@language{fr}{french}%
34 \smg@register@language{he}{hebrew}
35 \smg@register@language{hu}{hungarian}
36 \smg@register@language{id}{indonesian}
37 \smg@register@language{ms}{malay}
38 \smg@register@language{nn}{nynorsk}
39 \smg@register@language{pt}{portuguese}
40 \smg@register@language{ru}{russian}
41 \smg@register@language{uk}{ukrainian}
42 \smg@register@language{en}{english}
43 \smg@register@language{es}{spanish}
44 \smg@register@language{sq}{albanian}
45 \smg@register@language{bg}{bulgarian}
46 \smg@register@language{ca}{catalan}
47 \smg@register@language{hr}{croatian}
48 \smg@register@language{cs}{czech}
49 \smg@register@language{da}{danish}
50 \smg@register@language{nl}{dutch}
51 \smg@register@language{eo}{esperanto}
52 \smg@register@language{et}{estonian}
53 \smg@register@language{fi}{finnish}
54 \smg@register@language{ka}{georgian}
55 \smg@register@language{el}{greek}
56 \smg@register@language{is}{icelandic}
57 \smg@register@language{it}{italian}
58 \smg@register@language{la}{latin}
59 \smg@register@language{no}{norsk}
60 \smg@register@language{pl}{polish}
61 \smg@register@language{sr}{serbian}
62 \smg@register@language{sk}{slovak}
63 \smg@register@language{sl}{slovenian}
64 \smg@register@language{sv}{swedish}
65 \smg@register@language{th}{thai}
66 \smg@register@language{tr}{turkish}
67 \smg@register@language{vi}{vietnamese}
68 \smg@register@language{cy}{welsh}
69 \smg@register@language{hi}{hindi}
```

3.3 Language Bindings

modsig   The `modsig` environment is just a layer over the `module` environment. We also redefine macros that may occur in module signatures so that they do not create markup.

```
70 \newenvironment{modsig}[2][]{%
71 \def\@test{#1}\ifx\@test\@empty\begin{module}[id=#2]\else\begin{module}[id=#2,#1]\fi
72 \renewcommand\symtest[3][]{}
73 \renewcommand\abbrtest[3][]{}}
74 {\end{module}}
75 ⟨∗sty | ltxml⟩
76 ⟨ltxml⟩');
```

modnl:*

```
77 ⟨∗sty⟩
78 \addmetakey{modnl}{load}
79 \addmetakey*{modnl}{title}
80 \addmetakey*{modnl}{creators}
81 \addmetakey*{modnl}{contributors}
82 ⟨/sty⟩
83 ⟨∗ltxml⟩
84 DefKeyVal('modnl','title','Semiverbatim');
85 DefKeyVal('modnl','load','Semiverbatim');
86 DefKeyVal('modnl','creators','Semiverbatim');
87 DefKeyVal('modnl','contributors','Semiverbatim');
88 ⟨/ltxml⟩
```

modnl   The `modnl` environment is just a layer over the `module` environment with the keys and language suitably adapted.

```
89 ⟨∗sty⟩
90 \newenvironment{modnl}[3][]{\metasetkeys{modnl}{#1}%
91 \smg@select@language{#3}%
92 \def\@test{#1}\ifx\@test\@empty\begin{module}[id=#2.#3]\else\begin{module}[id=#2.#3,#1]\fi%
93 \if@langfiles\importmodule[load=#2,ext=tex]{#2}\else
94 \ifx\modnl@load\@empty\importmodule{#2}\else\importmodule[ext=tex,load=\modnl@load]{#2}\fi%
95 \fi}
96 {\end{module}}
97 ⟨/sty⟩
98 ⟨∗ltxml⟩
99 DefEnvironment('{modnl} OptionalKeyVals:modnl {}{}',
100        "<omdoc:theory "
101      . 'xml:id="#2.#3">'
102      .   "?&defined(&GetKeyVal(#1,'creators'))(<dc:creator>&GetKeyVal(#1,'creators')</dc:cr
103      .   "?&defined(&GetKeyVal(#1,'title'))(<dc:title>&GetKeyVal(#1,'title')</dc:title>)()"
104      .   "?&defined(&GetKeyVal(#1,'contributors'))(<dc:contributor>&GetKeyVal(#1,'contribut
105      .   "#body"
106      . "</omdoc:theory>",
107   afterDigestBegin=>sub {
108     my ($stomach, $whatsit) = @_;
```

```
109    my $keyval = $whatsit->getArg(1);
110    my $signature = ToString($whatsit->getArg(2));
111    if ($keyval) {
112      # If we're not given load, AND the langfiles option is in effect,
113      # default to #2
114      if ((! $keyval->getValue('load')) && (LookupValue('smultiling_langfiles'))) {
115        $keyval->setValue('load',$signature); }
116      # Always load a TeX file
117      $keyval->setValue('ext','tex'); }
118    importmoduleI($stomach,$whatsit)});
```
119 ⟨/ltxml⟩
120 ⟨ltxml⟩ *1;*