

`smglom.cls/sty`: Semantic Multilingual Glossary for Math

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

April 19, 2015

Abstract

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

This package supplies an infrastructure for writing OMDoc glossary entries.

Contents

1	Introduction	2
2	The User Interface	2
2.1	Package and Class Options	2
3	Implementation: The SMGloM Class	3
3.1	Class Options	3
3.2	For Module Definitions	4
3.3	For Language Bindings	6
3.4	Authoring States	6
3.5	Shadowing of repositories	6

1 Introduction

2 The User Interface

2.1 Package and Class Options

`smglom.cls` accepts all options of the `omdoc.cls` and `article.cls` and just passes them on to these.

3 Implementation: The SMGloM Class

3.1 Class Options

To initialize the `smglom` class, we pass on all options to `omdoc.cls`

```
1 <*cls>
2 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}}
3 \ProcessOptions
4 </cls>
5 <*ltxml.cls | ltxml.sty>
6 # -*- CPERL -*-
7 package LaTeXML::Package::Pool;
8 use strict;
9 use warnings;
10 use LaTeXML::Package;
11
12 DeclareOption(undef,sub {PassOptions('article','cls',ToString(Digest(T_CS('\CurrentOption'))));
13 ProcessOptions();
14 </ltxml.cls | ltxml.sty>
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
15 <*cls>
16 \LoadClass{omdoc}
17 \RequirePackage{smglom}
18 </cls>
19 <*sty>
20 \RequirePackage{amstext}
21 \RequirePackage{modules}
22 \RequirePackage{dcm}
23 \RequirePackage{statements}
24 \RequirePackage{sproof}
25 \RequirePackage{cmath}
26 \RequirePackage[langfiles]{smultiling}
27 \RequirePackage{presentation}
28 \RequirePackage{amsfonts}
29 </sty>
30 <*ltxml.cls>
31 LoadClass('omdoc');
32 RequirePackage('smglom');
33 </ltxml.cls>
34 <*ltxml.sty>
35 RequirePackage('amstext');
36 RequirePackage('modules');
37 RequirePackage('dcm');
38 RequirePackage('statements');
39 RequirePackage('sproof');
40 RequirePackage('cmath');
41 RequirePackage('smultiling',options => ['langfiles']);
42 RequirePackage('presentation');
```

```

43 RequirePackage('amsfonts');
44 </ltxml.sty>

```

3.2 For Module Definitions

`\gimport` just a shortcut, we have a starred and unstarred version, the first one is conservative.

```

45 <*sty>
46 \def\gimport{\@ifstar\gimport@star\gimport@nostar}
47 \newcommand\gimport@star[2][\def\@test{#1}\edef\mh@repos{\mh@currentrepos}%
48 \ifx\@test\@empty\importmhmodule[conservative,repos=\mh@repos,ext=tex,path=#2]{#2}%
49 \else\importmhmodule[conservative,repos=#1,ext=tex,path=#2]{#2}\fi%
50 \mhcurrentrepos\mh@repos\ignorespaces}
51 \newcommand\gimport@nostar[2][\def\@test{#1}\edef\mh@repos{\mh@currentrepos}%
52 \ifx\@test\@empty\importmhmodule[repos=\mh@repos,ext=tex,path=#2]{#2}%
53 \else\importmhmodule[repos=#1,ext=tex,path=#2]{#2}\fi%
54 \mhcurrentrepos\mh@repos\ignorespaces}
55 </sty>
56 <ltxml.sty>
57 DefMacro('gimport',' \@ifstar\gimport@star\gimport@nostar');
58 DefMacro('@gimport@star[]{}','\gimport[conservative=true,ext=tex,path=#2]{#1}{#2}');
59 DefMacro('@gimport@nostar[]{}','\gimport[conservative=false,ext=tex,path=#2]{#1}{#2}');
60 DefConstructor('\gimport OptionalKeyVals:importmhmodule {}{}',
61     "<omdoc:imports "
62     . "from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(\##2' "
63     . "conservative='&GetKeyVal(#1,'conservative')'/>",
64     afterDigest => \gimportI);

```

To make this work we need a sub that sets the respective values.

```

65 sub gimportI {
66   my ($stomach,$whatsit) = @_;
67   my $keyval = $whatsit->getArg(1);
68   my $repos = ToString($whatsit->getArg(2));
69   my $name = $whatsit->getArg(3);
70   if ($repos) {
71     $keyval->setValue('repos',$repos); }
72   else {
73     $keyval->setValue('repos',LookupValue('current_repos')); }
74   # Mystery: Why does $whatsit->setArgs($keyval,$name) raise a warning for
75   #           "odd numbers" in hash assignment? Workaround for now!
76   $$whatsit{args}[1] = $name; # Intention: $whatsit->setArg(2,$name);
77   undef $$whatsit{args}[2]; # Intention: $whatsit->deleteArg(3);
78   importMHmoduleI($stomach,$whatsit);
79   return; }##$
80 </ltxml.sty>

```

`guse` just a shortcut

```

81 <*sty>
82 \newcommand\guse[2][\def\@test{#1}%

```

```

83 \edef\mh@@repos{\mh@currentrepos}%
84 \ifx\@test\@empty\usemhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
85 \else\usemhmodule[repos=#1,ext=tex,path=#2]{#2}\fi
86 \mhcurrentrepos\mh@@repos\ignorespaces}
87 \</sty>
88 \<*ltxml.sty>
89 DefMacro('guse[]{}', 'g@use[ext=tex,path=#2]{#1}{#2}');
90 DefConstructor('g@use OptionalKeyVals:importmhmodule {} {}',
91   "<omdoc:uses from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(\##2
92   afterDigest => \&gimportI);
93 \</ltxml.sty>

```

gadopt just a shortcut

```

94 \<*sty>
95 \newcommand\gadopt[2][]{\def\@test{#1}%
96 \edef\mh@@repos{\mh@currentrepos}%
97 \ifx\@test\@empty\adoptmhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
98 \else\adoptmhmodule[repos=#1,ext=tex,path=#2]{#2}\fi
99 \mhcurrentrepos\mh@@repos\ignorespaces}
100 \</sty>
101 \<*ltxml.sty>
102 DefMacro('gadopt[]{}', 'g@adopt[ext=tex,path=#2]{#1}{#2}');
103 DefConstructor('g@adopt OptionalKeyVals:importmhmodule {} {}',
104   "<omdoc:adopts from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(\##2
105   afterDigest => \&gimportI);
106 \</ltxml.sty>

```

*nym

```

107 \<*sty>
108 \newcommand\hypernym[3][]{\if@importing\else\par\noindent #2 is a hypernym of #3\fi}
109 \newcommand\hyponym[3][]{\if@importing\else\par\noindent #2 is a hyponym of #3\fi}
110 \newcommand\meronym[3][]{\if@importing\else\par\noindent #2 is a meronym of #3\fi}
111 \</sty>
112 \<*ltxml.sty>
113 DefConstructor('hypernym [] {}{}', "");
114 DefConstructor('hyponym [] {}{}', "");
115 DefConstructor('meronym [] {}{}', "");
116 \</ltxml.sty>

```

EdN:1

\MSC to define the Math Subject Classification,¹

```

117 \<*sty>
118 \newcommand\MSC[1]{\if@importing\else MSC: #1\fi}
119 \</sty>
120 \<*ltxml.sty>
121 DefConstructor('MSC{}', "");
122 \</ltxml.sty>

```

¹EDNOTE: MK: what to do for the LaTeXML side?

3.3 For Language Bindings

Here we adapt the `smultiling` functionality to the special situation, where the module and file names are identical by design.

gviewsig The `gviewsig` environment is just a layer over the `viewsig` environment with the keys suitably adapted.

```
123 <ltxml.sty>RawTeX( '
124 <*sty | ltxml.sty>
125 \newenvironment{gviewsig}[4] [] {\def\test{#1}\ifx\@test\@empty%
126 \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}\else
127 \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}\fi}
128 {\end{mhviewsig}}}
```

gviewnl The `gve` environment is just a layer over the `viewnl` environment with the keys suitably adapted.

```
129 \newenvironment{gviewnl}[5] [] {\def\@test{#1}\ifx\@test\@empty%
130 \begin{mhviewnl}[frompath=#4,topath=#5]{#2}{#3}{#4}{#5}\else%
131 \begin{mhviewnl}[#1,frompath=#4,topath=#5]{#2}{#3}{#4}{#5}\fi}
132 {\end{mhviewnl}}
133 </sty | ltxml.sty>
134 <ltxml.sty>');;
```

3.4 Authoring States

We add a key to the module environment.

```
135 <*sty>
136 \addmetakey{module}{state}
137 </sty>
138 <*ltxml.sty>
139 DefKeyVal('modnl','state','Semiverbatim');
140 </ltxml.sty>
```

3.5 Shadowing of repositories

\repos@macro `\repos@macro` parses a GitLab repository name `<group>/<name>` and creates an internal macro name from that, which will be used

```
141 <*sty>
142 \def\repos@macro#1/#2;{#1@shadows@#2}
```

\shadow `\shadow{<orig>}{<fork>}` declares a that the private repository `<fork>` shadows the MathHub repository `<orig>`. Internally, it simply defines an internal macro with the shadowing information.

```
143 \def\shadow#1#2{\@namedef{\repos@macro#1;}{#2}}
144 </sty>
145 <*ltxml.sty>
146 DefConstructor('\shadow{}{}','');
147 </ltxml.sty>
```

`\MathHubPath` `\MathHubPath{⟨repos⟩}` computes the path of the fork that shadows the MathHub repository `⟨repos⟩` according to the current `\shadow` specification. The computed path can be used for loading modules from the private version of `⟨repos⟩`.

```

148 ⟨*sty⟩
149 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}
150 ⟨/sty⟩
151 ⟨*ltxml.sty⟩
152 DefConstructor('MathHubPath{','');
153 ⟨/ltxml.sty⟩

```