

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

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

November 22, 2015

## Abstract

The `smglom` package is part of the  $\text{\LaTeX}$  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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>The User Interface</b>	<b>2</b>
2.1	Package and Class Options . . . . .	2
<b>3</b>	<b>Implementation: The SMGloM Class</b>	<b>3</b>
3.1	Class Options . . . . .	3
3.2	For Module Definitions . . . . .	4
3.3	For Language Bindings . . . . .	6
3.4	Authoring States . . . . .	7
3.5	Shadowing of repositories . . . . .	7

## **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

The general preamble for L<sup>A</sup>T<sub>E</sub>XML(class and package)

```
1 <*ltxml.cls | ltxml.sty>
2 # -*- CPERL -*-
3 package LaTeXML::Package::Pool;
4 use strict;
5 use warnings;
6 use LaTeXML::Package;
7 </ltxml.cls | ltxml.sty>
```

#### 3.1 Class Options

To initialize the `smglom` class, we pass on all options to `omdoc.cls` as well as the `stex` and `smglom` packages.

```
8 <*cls>
9 \DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}
10                                     \PassOptionsToPackage{\CurrentOption}{stex}
11                                     \PassOptionsToPackage{\CurrentOption}{smglom}}
12 \ProcessOptions
13 </cls>
14 <*ltxml.cls>
15 \DeclareOption(undef,sub {PassOptions('omdoc','cls', ToString(Digest(T_CS('\CurrentOption'))));
16                                     PassOptions('stex', 'sty', ToString(Digest(T_CS('\Curr
17                                     PassOptions('smglom','sty',ToString(Digest(T_CS('\Curr
18 \ProcessOptions();
19 </ltxml.cls>
```

We load `omdoc.cls`, the `smglom` package that provides the SMGloM-specific functionality<sup>1</sup>, and the `stex` package to allow OMDoc compatibility.

```
20 <*cls>
21 \LoadClass{omdoc}
22 \RequirePackage{smglom}
23 \RequirePackage{stex}
24 \RequirePackage{amstext}
25 \RequirePackage{amsfonts}
26 </cls>
27 <*ltxml.cls>
28 \LoadClass('omdoc');
29 \RequirePackage('stex');
30 \RequirePackage('smglom');
31 \RequirePackage('amstext');
32 \RequirePackage('amsfonts');
33 </ltxml.cls>
```

Now we do the same thing for the package; first the options, which we just pass on to the `stex` package.

---

<sup>1</sup>EdNOTE: MK:describe that above

```

34 <*sty>
35 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{statements}
36                                     \PassOptionsToPackage{\CurrentOption}{structview}
37                                     \PassOptionsToPackage{\CurrentOption}{smultiling}}
38 \ProcessOptions
39 </sty>
40 <*ltxml.sty>
41 \DeclareOption(undef,sub {PassOptions('statements','sty',ToString(Digest(T_CS('\CurrentOption'))
42                               PassOptions('structview','sty',ToString(Digest(T_CS('\CurrentOption'))));
43                               PassOptions('smultiling','sty',ToString(Digest(T_CS('\CurrentOption')))); });
44 \ProcessOptions();
45 </ltxml.sty>

```

We load `omdoc.cls`, and the desired packages. For the L<sup>A</sup>T<sub>E</sub>XML bindings, we make sure the right packages are loaded.

```

46 <*sty>
47 \RequirePackage{statements}
48 \RequirePackage[langfiles]{smultiling}
49 \RequirePackage{structview}
50 </sty>
51 <*ltxml.sty>
52 \RequirePackage('statements');
53 \RequirePackage('smultiling',options => ['langfiles']);
54 \RequirePackage('structview');
55 </ltxml.sty>

```

### 3.2 For Module Definitions

`\gimport` Just a shortcut, we have a starred and unstarred version, the first one is conservative. For example, if we execute:

```
\gimport[smglom/numberfields]{naturalnumbers}
```

First we are redirected to `\@gimport@nostar`, we store the `smglom/numberfields` (*the repo's path*) in `\@test`, then store `\mh@currentrepos` (*current directory*) in `\mh@repos`. If no repo's path is offered, that means the module to import is under the same directory, so we let `repos=\mh@repos` and pass bunch of parameters to `\importmhmodule`, which is defined in `module.sty`. If there's a repo's path, then we let `repos=\@test`. Finally we use `\mhcurrentrepos` (defined in `module.sty`) to change the `\mh@currentrepos`.

```

56 <*sty>
57 \def\gimport{\@ifstar\@gimport@star\@gimport@nostar}%
58 \newrobustcmd\@gimport@star[2][]{%
59   \def\@test{#1}%
60   \edef\mh@repos{\mh@currentrepos}%
61   \ifx\@test\empty%
62     \importmhmodule[conservative,repos=\mh@repos,ext=tex,path=#2]{#2}%

```

```

63 \else%
64 \importmhmodule[conservative,repos=#1,ext=tex,path=#2]{#2}%
65 \fi%
66 \mhcurrentrepos{\mh@@repos}%
67 \ignorespaces%
68 }%
69 \newrobustcmd\@gimport@nostar[2][]{%
70 \def\@test{#1}%
71 \edef\mh@@repos{\mh@currentrepos}%
72 \ifx\@test\@empty%
73 \importmhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
74 \else%
75 \importmhmodule[repos=#1,ext=tex,path=#2]{#2}%
76 \fi%
77 \mhcurrentrepos{\mh@@repos}%
78 \ignorespaces%
79 }%
80 \</sty>
81 \<*lxml.sty>
82 DefMacro('gimport', '\@ifstar\@gimport@star\@gimport@nostar');
83 DefMacro('@gimport@star[]{}', '\gimport[conservative=true,ext=tex,path=#2]{#1}{#2}');
84 DefMacro('@gimport@nostar[]{}', '\gimport[conservative=false,ext=tex,path=#2]{#1}{#2}');
85 DefConstructor('\gimport OptionalKeyVals:importmhmodule {}{}',
86 " <omdoc:imports "
87 . "from=?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(###2' "
88 . "conservative=?&GetKeyVal(#1,'conservative'))"/>",
89 afterDigest => \&gimportI);

```

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

```

90 sub gimportI {
91 my ($stomach,$whatsit) = @_;
92 my $keyval = $whatsit->getArg(1);
93 my $repos = ToString($whatsit->getArg(2));
94 my $name = $whatsit->getArg(3);
95 if ($repos) {
96 $keyval->setValue('repos',$repos); }
97 else {
98 $keyval->setValue('repos',LookupValue('current_repos')); }
99 # Mystery: Why does $whatsit->setArgs($keyval,$name) raise a warning for
100 # "odd numbers" in hash assignment? Workaround for now!
101 $$whatsit{args}[1] = $name; # Intention: $whatsit->setArg(2,$name);
102 undef $$whatsit{args}[2]; # Intention: $whatsit->deleteArg(3);
103 importMHmoduleI($stomach,$whatsit);
104 return; }##$
105 \</lxml.sty>

```

guse just a shortcut

```

106 \<*sty>
107 \newrobustcmd\guse[2][]{%
108 \def\@test{#1}%

```

```

109 \edef\mh@@repos{\mh@currentrepos}%
110 \ifx\@test\@empty%
111   \usemhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
112 \else%
113   \usemhmodule[repos=#1,ext=tex,path=#2]{#2}%
114 \fi%
115 \mhcurrentrepos{\mh@@repos}%
116 \ignorespaces%
117 }%
118 </sty>
119 <*ltxml.sty>
120 DefMacro('guse[]{}', 'g@use[ext=tex,path=#2]{#1}{#2}');
121 DefConstructor('g@use OptionalKeyVals:importmhmodule {} {}',
122   "<omdoc:uses from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))(###2
123   afterDigest => \&gimportI);
124 </ltxml.sty>

```

\*nym

```

125 <*sty>
126 \newrobustcmd\hypernym[3] [] {\if@importing\else\par\noindent #2 is a hypernym of #3\fi}%
127 \newrobustcmd\hyponym[3] [] {\if@importing\else\par\noindent #2 is a hyponym of #3\fi}%
128 \newrobustcmd\meronym[3] [] {\if@importing\else\par\noindent #2 is a meronym of #3\fi}%
129 </sty>
130 <*ltxml.sty>
131 DefConstructor('hypernym [] {}{}', "");
132 DefConstructor('hyponym [] {}{}', "");
133 DefConstructor('meronym [] {}{}', "");
134 </ltxml.sty>

```

EdN:2

\MSC to define the Math Subject Classification, <sup>2</sup>

```

135 <*sty>
136 \newrobustcmd\MSC[1] {\if@importing\else MSC: #1\fi}%
137 </sty>
138 <*ltxml.sty>
139 DefConstructor('MSC{}', "");
140 </ltxml.sty>

```

### 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 `mhviewsig` environment with the keys suitably adapted.

```

141 <ltxml.sty>RawTeX('
142 <*sty | ltxml.sty>
143 \newenvironment{gviewsig}[4] [] {%

```

---

<sup>2</sup>EdNOTE: MK: what to do for the LaTeXML side?

```

144 \def\test{#1}%
145 \ifx\@test\@empty%
146   \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}%
147   \else%
148     \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}%
149   \fi%
150 }{%
151 \end{mhviewsig}%
152 }%

```

**gviewnl** The **gviewnl** environment is just a layer over the **mhviewnl** environment with the keys suitably adapted.

```

153 \newenvironment{gviewnl}[5][]{%
154 \def\@test{#1}\ifx\@test\@empty%
155   \begin{mhviewnl}[frompath=#3,topath=#4]{#2}{#3}{#4}{#5}%
156   \else%
157     \begin{mhviewnl}[#1,frompath=#3,topath=#4]{#2}{#3}{#4}{#5}%
158   \fi%
159 }{%
160 \end{mhviewnl}%
161 }%
162 \</sty | ltxml.sty>
163 \<ltxml.sty>'');

```

EdN:3

```

\gincludeview 3
164 \<*sty>
165 \newcommand\gincludeview[2][]{}%
166 \</sty>
167 \<*ltxml.sty>
168 DefConstructor('gincludeview[]{}','');
169 \</ltxml.sty>

```

### 3.4 Authoring States

We add a key to the module environment.

```

170 \<*sty>
171 \addmetakey{module}{state}%
172 \</sty>
173 \<*ltxml.sty>
174 DefKeyVal('modnl','state','Semiverbatim');
175 \</ltxml.sty>

```

### 3.5 Shadowing of repositories

**\repos@macro** **\repos@macro** parses a GitLab repository name  $\langle group \rangle / \langle name \rangle$  and creates an internal macro name from that, which will be used

```

176 \<*sty>

```

---

<sup>3</sup>EDNOTE: This is fake for now, needs to be implemented and documented

```

177 \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.

178 \def\shadow#1#2{\@namedef{\repos@macro#1;}{#2}}%
179 \</sty>
180 \<*ltxml.sty>
181 DefConstructor('shadow{ }', '');
182 \</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>.

183 \<*sty>
184 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}%
185 \</sty>
186 \<*ltxml.sty>
187 DefConstructor('MathHubPath{ }', '');
188 \</ltxml.sty>

and finally, we need to terminate the file with a success mark for perl.
189 \<ltxml.sty | ltxml.cls>1;

```