

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

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

November 5, 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('C
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}{modules}}
36 \ProcessOptions
37 </sty>
38 <*ltxml.sty>
39 DeclareOption(undef,sub {PassOptions('modules','sty',ToString(Digest(T_CS('\CurrentOption'))));
40 ProcessOptions();
41 </ltxml.sty>

```

We load `omdoc.cls`, and the desired packages. For the  $\text{\LaTeX}$ XML bindings, we make sure the right packages are loaded.

```

42 <*sty>
43 \RequirePackage{modules-mh}
44 \RequirePackage[langfiles]{smultiling-mh}
45 \RequirePackage{structview-mh}
46 </sty>
47 <*ltxml.sty>
48 RequirePackage('modules');
49 RequirePackage('smultiling',options => ['langfiles']);
50 </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=`(*the repo's path*). Finally we use `\mhcurrentrepos` (defined in `module.sty`) to change the `\mh@currentrepos`.

```

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

```

```

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

```

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

```

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

```

**guse** just a shortcut

```

101 \<sty>
102 \newrobustcmd\guse[2][]{%
103   \def\@test{#1}%
104   \edef\mh@@repos{\mh@currentrepos}%
105   \ifx\@test\@empty%
106     \usemhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
107   \else%
108     \usemhmodule[repos=#1,ext=tex,path=#2]{#2}%

```

```

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

```

\*nym

```

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

```

EdN:2

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

```

130 \<*sty>
131 \newrobustcmd\MSC[1] {\if@importing\else MSC: #1\fi}%
132 \</sty>
133 \<*ltxml.sty>
134 DefConstructor('\MSC{}', "");
135 \</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 `viewsig` environment with the keys suitably adapted.

```

136 \<ltxml.sty>RawTeX(
137 \<*sty | ltxml.sty>
138 \newenvironment{gviewsig}[4] [] {%
139   \def\test{#1}%
140   \ifx\@test\@empty%
141     \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}%
142   \else%
143     \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}%

```

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

```

144 \fi%
145 }{%
146 \end{mhviewsig}%
147 }%

```

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

```

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

```

### 3.4 Authoring States

We add a key to the module environment.

```

159 \<sty>
160 \addmetakey{module}{state}%
161 \</sty>
162 \<ltxml.sty>
163 DefKeyVal('modnl','state','Semiverbatim');
164 \</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

```

165 \<sty>
166 \def\repos@macro#1/#2;{#1@shadows@#2}%

```

**\shadow** **\shadow{ $\langle orig \rangle$ { $\langle fork \rangle$ }}** declares a that the private repository  $\langle fork \rangle$  shadows the MathHub repository  $\langle orig \rangle$ . Internally, it simply defines an internal macro with the shadowing information.

```

167 \def\shadow#1#2{\@namedef{\repos@macro#1;}{#2}}%
168 \</sty>
169 \<ltxml.sty>
170 DefConstructor('\shadow{}{}','');
171 \</ltxml.sty>

```

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

```

172 <*sty>
173 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}%
174 </sty>
175 <*ltxml.sty>
176 DefConstructor('\MathHubPath{}', '');
177 </ltxml.sty>

```

and finally, we need to terminate the file with a success mark for perl.

```

178 <ltxml.sty | ltxml.cls>1;

```