

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

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

August 5, 2014

## Abstract

The `smglom` 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).

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 . . . . .	5
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<sup>A</sup>T<sub>E</sub>XML 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');
42 RequirePackage('presentation');
```

```

43 \RequirePackage('amsfonts');
44 \ltxml.sty

```

## 3.2 For Module Definitions

`gimport` just a shortcut

```

45 \ltsty
46 \newcommand\gimport[2] [] {\def\@test{#1}%
47 \edef\mh@@repos{\mh@currentrepos}%
48 \ifx\@test\@empty\importmhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
49 \else\importmhmodule[repos=#1,ext=tex,path=#2]{#2}\fi
50 \mhcurrentrepos\mh@@repos\ignorespaces}
51 \ltsty
52 \ltxml.sty
53 \DefMacro('gimport[]{}', '\g@import[ext=tex,path=#2]{#1}{#2}');
54 \DefConstructor('g@import OptionalKeyVals:importmhmodule {}{}',
55 " <omdoc:imports from='?'&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')) )\
56 afterDigest => \&gimportI);

```

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

```

57 sub gimportI {
58 my ($stomach,$whatsit) = @_;
59 my $keyval = $whatsit->getArg(1);
60 my $repos = ToString($whatsit->getArg(2));
61 my $name = $whatsit->getArg(3);
62 if ($repos) {
63 $keyval->setValue('repos',$repos); }
64 else {
65 $keyval->setValue('repos',LookupValue('current_repos')); }
66 # Mystery: Why does $whatsit->setArgs($keyval,$name) raise a warning for
67 # "odd numbers" in hash assignment? Workaround for now!
68 $$whatsit{args}[1] = $name; # Intention: $whatsit->setArg(2,$name);
69 undef $$whatsit{args}[2]; # Intention: $whatsit->deleteArg(3);
70 importMHmoduleI($stomach,$whatsit);
71 return; }##$
72 \ltxml.sty

```

`guse` just a shortcut

```

73 \ltsty
74 \newcommand\guse[2] [] {\def\@test{#1}%
75 \edef\mh@@repos{\mh@currentrepos}%
76 \ifx\@test\@empty\usemhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
77 \else\usemhmodule[repos=#1,ext=tex,path=#2]{#2}\fi
78 \mhcurrentrepos\mh@@repos\ignorespaces}
79 \ltsty
80 \ltxml.sty
81 \DefMacro('guse[]{}', '\g@use[ext=tex,path=#2]{#1}{#2}');
82 \DefConstructor('g@use OptionalKeyVals:importmhmodule {}{}',
83 " <omdoc:uses from='?'&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')) )\##2

```

```

84 afterDigest => \&gimportI);
85 \ltxml.sty>

gadopt just a shortcut
86 \*sty>
87 \newcommand\gadopt[2] [] {\def\@test{#1}%
88 \edef\mh@@repos{\mh@currentrepos}%
89 \ifx\@test\@empty\adoptmhmodule[repos=\mh@@repos,ext=tex,path=#2]{#2}%
90 \else\adoptmhmodule[repos=#1,ext=tex,path=#2]{#2}\fi
91 \mhcurrentrepos\mh@@repos\ignorespaces}
92 \</sty>
93 \*ltxml.sty>
94 DefMacro('gadopt [] {}', 'g@adopt[ext=tex,path=#2]{#1}{#2}');
95 DefConstructor('g@adopt OptionalKeyVals:importmhmodule {} {}',
96 " <omdoc:adopts from='?&GetKeyVal(#1,'load')(&canonical_omdoc_path(&GetKeyVal(#1,'load')))> \#
97 afterDigest => \&gimportI);
98 \ltxml.sty>

```

\*nym

```

99 \*sty>
100 \newcommand\hypernym[3] [] {\if@importing\else\par\noindent #2 is a hypernym of #3\fi}
101 \newcommand\hyponym[3] [] {\if@importing\else\par\noindent #2 is a hyponym of #3\fi}
102 \newcommand\meronym[3] [] {\if@importing\else\par\noindent #2 is a meronym of #3\fi}
103 \</sty>
104 \*ltxml.sty>
105 DefConstructor('hypernym [] {}{}', "");
106 DefConstructor('hyponym [] {}{}', "");
107 DefConstructor('meronym [] {}{}', "");
108 \ltxml.sty>

```

EdN:1

\MSC to define the Math Subject Classification, <sup>1</sup>

```

109 \*sty>
110 \newcommand\MSC[1] {\if@importing\else MSC: #1\fi}
111 \</sty>
112 \*ltxml.sty>
113 DefConstructor('MSC{}', "");
114 \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.

```
115 \ltxml.sty>RawTeX('
```

---

<sup>1</sup>EdNOTE: MK: what to do for the LaTeXML side?

```

116 <*sty | ltxml.sty>
117 \newenvironment{gviewsig}[4] [] {\def\test{#1}\ifx\@test\@empty%
118 \begin{mhviewsig}[frompath=#3,topath=#4]{#2}{#3}{#4}\else
119 \begin{mhviewsig}[frompath=#3,topath=#4,#1]{#2}{#3}{#4}\fi}
120 {\end{mhviewsig}}

```

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

```

121 \newenvironment{gviewnl}[5] [] {\def\@test{#1}\ifx\@test\@empty%
122 \begin{mhviewnl}[frompath=#4,topath=#5]{#2}{#3}{#4}{#5}\else%
123 \begin{mhviewnl}[#1,frompath=#4,topath=#5]{#2}{#3}{#4}{#5}\fi
124 \smul@select@language{#3}}
125 {\end{mhviewnl}}
126 </sty | ltxml.sty>
127 <ltxml.sty>');

```

### 3.4 Authoring States

We add a key to the module environment.

```

128 <*sty>
129 \addmetakey{module}{state}
130 </sty>
131 <*ltxml.sty>
132 DefKeyVal('modnl','state','Semiverbatim');
133 </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

```

134 <*sty>
135 \def\repos@macro#1/#2;{#1@shadows@#2}

```

**\shadow** **\shadow** $\{\langle orig \rangle\}$ **\shadow** $\{\langle fork \rangle\}$  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.

```

136 \def\shadow#1#2{\@namedef{\repos@macro#1;}{#2}}
137 </sty>
138 <*ltxml.sty>
139 DefConstructor('\shadow{}{}','');
140 </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  $\langle repos \rangle$ .

```

141 <*sty>
142 \def\MathHubPath#1{\@ifundefined{\repos@macro#1;}{#1}{\@nameuse{\repos@macro#1;}}}
143 </sty>

```

```
144 <*|xml.sty>  
145 DefConstructor('MathHubPath{ }', '');  
146 </|xml.sty>
```