Slides and Course Notes for Jacobs University*

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

January 28, 2012

Abstract

We present a document class from which we can generate both course slides and course notes in a transparent way. Furthermore, we present a set of LaTeXML bindings for these, so that we can also generate OMDoc-based course materials, e.g. for inclusion in the ActiveMath system.

Contents

*Version ? (last revised ?)

1 Introduction

This Document class is derived from `beamer.cls`, specializes it with Jacobs stuff and adds a notes version that is more suited to printing than the one supplied by `beamer.cls`.

2 The User Interface

2.1 Package Options

EdNote:1

The `mikoslides` class takes a variety of class options:[1]

`showmeta`
- q`showmeta`. If this is set, then the metadata keys are shown (see [Koh10] for details and customization options).

`slides`
`a`
- The options `slides`nd notes`notes` switch between slides mode and notes mode (see Section 2.2).

`sectocframes`
- If the option `sectocframes` is given, then special frames with section table of contents are producedheaders [2]

EdNote:2

2.2 Notes and Slides

2.3 Header and Footer Lines

2.4 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: ⚠

2.5 Front Matter, Titles, etc

2.6 Miscellaneous

3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the sTEXTRAC [Ste].

1. the class should be divided into concerns. [Ste], issue 1684

4 The Implementation

The `mikoslides` package generates two files: the LaTeX package (all the code between ⟨*package⟩ and ⟨/package⟩) and the LaTeXML bindings (between ⟨*ltxml⟩ and ⟨/ltxml⟩). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

[1] EdNote: leaving out noproblems for the momeent until we decide what to do with it.
[2] EdNote: document the functionality

4.1 Initialization and Class Options

For the LaTeXML bindings, we make sure the right perl packages are loaded.

```
⟨*ltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
⟨/ltxml⟩
```

For LaTeX we define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` the appropriate packages.

```
⟨*cls⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\newif\ifnotes\notesfalse
\newif\ifsectocframes\sectocframesfalse
\newif\ifproblems\problemstrue
\DeclareOption{notes}{\notestrue}
\DeclareOption{slides}{\notesfalse}
\DeclareOption{noproblems}{\problemsfalse}
\DeclareOption{sectocframes}{\sectocframestrue}
\ifnotes
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}}
\else
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{beamer}}
\fi
\ProcessOptions
⟨/cls⟩
⟨*ltxml⟩
RawTeX('\newif\ifnotes\notesfalse');
RawTeX('\newif\ifproblems\problemsfalse');
⟨/ltxml⟩
```

Depending on the options, we either load the `article`-based `omdoc` or the `beamer` class. In the first case, we also have to make the `beamer`-specific things available to `article` via the `beamerarticle` package. We use options to avoid loading theorem-like environments, since we want to use our own from the STeX packages.

```
⟨*cls⟩
\ifnotes
\LoadClass{omdoc}
\RequirePackage{a4wide}
\RequirePackage{marginnote}
\RequirePackage[notheorems,noamsthm]{beamerarticle}
\else
\LoadClass[notheorems,noamsthm,10pt]{beamer}
\newcounter{Item}
\newcounter{paragraph}
```

```
\newcounter{subparagraph}
\newcounter{Hfootnote}
\usetheme{Jacobs}
\fi
⟨/cls⟩
⟨*ltxml⟩
LoadClass('omdoc');
DefConstructor('\usetheme{}','');
⟨/ltxml⟩
```

EdNote:3

now, we load the remaining packages for both versions. [3]

```
⟨*cls⟩
\RequirePackage{stex}
\RequirePackage{latexml}
\RequirePackage{amssymb}
\RequirePackage{tikz}
\usepgflibrary{shapes}\usetikzlibrary{arrows}
\RequirePackage{url}
\RequirePackage{amsmath}
\RequirePackage{comment}
\RequirePackage{standalone}
⟨/cls⟩
⟨*ltxml⟩
RequirePackage('stex');
RequirePackage('latexml');
RequirePackage('amssymb');
RequirePackage('graphicx');
RequirePackage('tikz');
RequirePackage('amsmath');
⟨/ltxml⟩
```

4.2 Notes and Slides

We define the sizes of slides in the notes. Somehow, we cannot get by with the same here.

```
⟨*cls⟩
\newcounter{slide}
\newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
\newlength{\slideheight}\setlength{\slideheight}{9cm}
⟨/cls⟩
⟨*ltxml⟩
DefRegister('\slidewidth'     => Dimension('13.5cm'));
DefRegister('\slideheight'     => Dimension('9cm'));
⟨/ltxml⟩
```

For course notes, we define the `note` environment to be a no-operation otherwise we declare the `note` environment as a comment via the `comment` package.

[3] EdNote: MK: eventually (when tikz support is fully realized in LaTeXML) get rid of the standalone package

note

```
⟨*cls⟩
\ifnotes\renewenvironment{note}{}{}\else\excludecomment{note}\fi
⟨/cls⟩
⟨*ltxml⟩
DefEnvironment('{note}','#body');
⟨/ltxml⟩
```

the next step is to set up the slide boxes in `article` mode. We set up sizes and provide a box register for the frames and a counter for the slides.

```
⟨*cls⟩
\ifnotes
\newlength{\slideframewidth}\setlength{\slideframewidth}{2pt}
\newsavebox{\myframebox}
⟨/cls⟩
```

frame For the `frame` environment we construct a `lrbox` in the `\myframebox` register that we can later put into an `\fbox` so that it looks like a slide. Furthermore, we redefine the `itenize` environment so that it looks more like the one in `beamer` with `Jacobs` theme.

```
⟨*cls⟩
\renewenvironment{frame}[1][]%
{\stepcounter{slide}
\def\itemize@level{outer}
\def\itemize@outer{outer}
\def\itemize@inner{inner}
\renewcommand\metakeys@show@keys[2]{\marginnote{{\scriptsize ##2}}}
\renewenvironment{itemize}
{\ifx\itemize@level\itemize@outer\def\itemize@label{$\rhd$}\fi
 \ifx\itemize@level\itemize@inner\def\itemize@label{$\scriptstyle\rhd$}\fi
  \begin{list}
    {\itemize@label}
    {\setlength{\labelsep}{.3em}\setlength{\labelwidth}{.5em}\setlength{\leftmargin}{1.5em}}
  \edef\itemize@level{\itemize@inner}}
{\end{list}}
 \noindent\hfill\begin{lrbox}{\myframebox}
   \begin{minipage}{\slidewidth}\sf}%
   {\miko@slidelabel\end{minipage}\end{lrbox}%
 \begin{center}\fbox{\usebox\myframebox}\end{center}\hfill}
⟨/cls⟩
⟨*ltxml⟩
DefEnvironment('{frame}[]',
   "<omdoc:omgroup layout='slide'>"
  .  "#body\n"
  ."</omdoc:omgroup>\n\n",
 afterDigestBegin=>sub {
   $_[1]->setProperty(theory=>LookupValue('current_module')); });
⟨/ltxml⟩#$
```

the next step is to set up the slide boxes in `article` mode.

```
⟨*cls⟩
\renewcommand{\frametitle}[1]{{\Large\bf\sf\color{blue}{#1}}}
\fi
\makeindex
⟨/cls⟩
⟨*ltxml⟩
DefConstructor('\frametitle{}',
  "\n<omdoc:metadata><dc:title>#1</dc:title></omdoc:metadata>");
⟨/ltxml⟩
```

We start by giving the LaTeXML binding for the `frame` environment from the `beamer` class. The `note` environment is used to blend out text in the `slides` mode. It does not have a counterpart in OMDoc.

```
⟨*cls⟩
\ifproblems\newenvironment{problems}{}{}\else\excludecomment{problems}\fi
⟨/cls⟩
⟨*ltxml⟩
DefEnvironment('{problems}','#body');
⟨/ltxml⟩
```

EdNote:4 [4]

4.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

```
⟨*cls⟩
\newlength{\slidelogoheight}
\ifnotes\setlength{\slidelogoheight}{.4cm}\else\setlength{\slidelogoheight}{1cm}\fi
\newsavebox{\slidelogo}\sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{jacobs-logo}}
```

Now, we set up the copyright and licensing, the copyright remains with the author, but we use the Creative Commons Attribuition-ShareAlike license to strengthen den public domain. Here the problem is that we want a hyperref on the CC logo, if hyperref is loaded, and otherwise not. As hyperref is always loaded, we have to find out at the beginning of the document whether it is, set up a switch, and later in the footer line decide what to do.

```
\def\source{Michael Kohlhase}% customize locally
\def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}
\newsavebox{\cclogo}\sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
\newif\ifcchref\cchreffalse
\AtBeginDocument{\@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}}
\def\licensing{\ifcchref\href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
```

EdNote:5 Now, we set up the slide label for the `article` mode[5]

[4] EdNote: subtitle is difficult to model in DC metadata. I guess that we want to collect the subtitle into dc:title

[5] EdNote: see that we can use the themes for the slides some day. This is all fake.

\slidelabel

```
\newcommand{\miko@slidelabel}%
{\vbox to \slidelogoheight{\vss\hbox to \slidewidth%
{\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}}}
⟨/cls⟩
```

4.4 Colors and Highlighting

Now, we set up an infrastructure for highlighting phrases in slides. Note that we use content-oriented macros for highlighting rather than directly using color markup. The first thing to to is to adapt the green so that it is dark enough for most beamers

```
⟨*cls⟩
\AtBeginDocument{\definecolor{green}{rgb}{0,.5,0}\definecolor{purple}{cmyk}{.3,1,0,.17}}
```

We customize the `\defemph`, `\notemph`, and `\stDMemph` macros with colors for the use in the `statements` package. Furthermore we customize the `\@@lec` macro for the appearance of line end comments in `\lec`.

```
% \def\STpresent#1{\textcolor{blue}{#1}}
\def\defemph#1{{\textcolor{magenta}{#1}}}
\def\notemph#1{{\textcolor{magenta}{#1}}}
\def\stDMemph#1{{\textcolor{blue}{#1}}}
\def\@@lec#1{(\textcolor{green}{#1})}
⟨/cls⟩
⟨*ltxml⟩
#DefMacro('\defemph{}','{\textcolor{magenta}{#1}}');
#DefMacro('\notemph{}','{\textcolor{magenta}{#1}}');
⟨/ltxml⟩
```

I like to use the dangerous bend symbol for warnings, so we provide it here.

\textwarigrening

as the macro can be used quite often we put it into a box register, so that it is only loaded once.

```
⟨*cls⟩
\pgfdeclareimage[width=1.5em]{miko@dbend}{dangerous-bend}
\def\textwarning{\raisebox{-.05cm}{\pgfuseimage{miko@dbend}}\xspace}
⟨/cls⟩b
⟨*ltxml⟩
DefMacro('\textwarning',"");
⟨/ltxml⟩
```

4.5 Front Matter, Titles, etc

We need to redefine the frontmatter macros inherited from the `beamer` class, since there they take an optional argument.

```
⟨*ltxml⟩
DefMacro('\title[]{}', '\@add@frontmatter{ltx:title}{#1}');
DefMacro('\date[]{}', '\@add@frontmatter{ltx:date}[role=creation]{#1}');
DefMacro('\author[]{}', sub { andSplit(T_CS('\@author'),$_[1]); });#$
```

¡/ltxml¿

Now, we specialize the slide environment that we have implemented above or inherited from `seminar.cls` for some abbreviations, e.g. separator slides and title slides.

```
⟨*cls⟩
\newcommand\titleframe{\begin{frame}\titlepage\end{frame}}
\newenvironment{titleframewith}{\begin{frame}\titlepage}{\end{frame}}
\newenvironment{ttitle}{\begin{center}\LARGE\begin{tabular}{|c|}\hline}%
 {\\\hline\end{tabular}\end{center}\vspace{1ex minus 1ex}}
\newenvironment{ttitlejoint}[1]%
{\newbox\boxwith\setbox\boxwith\hbox{\begin{tabular}{c}{\em joint work with}\\#1\end{tabular}}%
\begin{center}\LARGE\begin{tabular}{c}\color{red}}%
{\\\box\boxwith\end{tabular}\end{center}%
\vspace{1ex minus 1ex}}
⟨/cls⟩
⟨*ltxml⟩
DefConstructor('\titleframe',"<omdoc:ignore>titleframe elided here</omdoc:ignore>");
DefEnvironment('{titleframewith}',
               "<omdoc:ignore>begin elided titleframe</omdoc:ignore>"
               . "#body"
              ."<omdoc:ignore>end elided titleframe</omdoc:ignore>");
DefEnvironment('{titleslide}',"");
DefEnvironment('{titleslide}',"<omdoc:omgroup>#body</omdoc:omgroup>");
DefEnvironment('{ttitle}', "\n<dc:title>#body</dc:title>");
⟨/ltxml⟩
%      Must be first command on slide to make positioning work.
⟨*cls⟩
\newcommand{\putgraphicsat}[3]{%
 \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}}
\newcommand{\putat}[2]{\begin{picture}(0,0)\put(#1){#2}\end{picture}}
⟨/cls⟩
⟨*ltxml⟩
⟨/ltxml⟩
⟨*cls⟩
\ifsectocframes
%\AtBeginChapter[]{\begin{frame}<beamer>\frametitle{Chapter Outline}\tableofcontents[currentsec
%\AtBeginSection[]{\begin{frame}<beamer>\frametitle{Section Outline}\tableofcontents[currentsec
%\AtBeginSubsection[]{\begin{frame}<beamer>\frametitle{Subsection Outline}\tableofcontents[curr
%\AtBeginSubsubsection[]{\begin{frame}<beamer>\frametitle{Subsubsection Outline}\tableofcontent
\def\at@begin@omgroup#1{\message{atbeginomgroup}\begin{frame}<beamer>\frametitle{Outline}\table
\fi
⟨/cls⟩
%
% \subsection{Miscellaneous}
%
% the following macro is only to work around problems in the |tikz| support in \latexml.
%    \begin{macrocode}
⟨*cls⟩
\newcommand\tikzinput[2][]{\input{#2}}
```

```
⟨/cls⟩
⟨*ltxml⟩
DefMacro('\tikzinput[] {}','\includegraphics[#1]{#2}');
⟨/ltxml⟩
```

We need to disregard the columns macros introduced by the `beamer` class

```
⟨*ltxml⟩
DefEnvironment('{columns}','#body');
DefEnvironment('{column}{}','#body');
⟨/ltxml⟩
```

We also need to deal with overlay specifications introduced by the `beamer` class.[6]

EdNote:6
EdNote:7
[7]

```
⟨*ltxml⟩
DefConstructor('\uncover','#1');
#Define a Beamer Overlay Parameter type
 DefParameterType('BeamerOverlay', sub {
   my ($gullet) = @_;
   my $tok = $gullet->readXToken;
   if (ref $tok && ToString($tok) eq '<') {
     $gullet->readUntil(T_OTHER('>'));
   } else {
     $gullet->unread($tok) if ref $tok;
     undef; }},
     reversion=> sub {
 (T_OTHER('<'), $_[0]->revert, T_OTHER('>'));
       });

#Take the "from" field of the overlay range
sub overlayFrom {
  return "" unless defined $_[0];
  my $overlay=ToString($_[0]); $overlay =~ /^(\d+)/; $1;}

#Reuse the CMP itemizations, only adjust the \item constructors.
DefMacro('\beamer@group@item[] OptionalBeamerOverlay IfBeginFollows', sub {
  my($gullet,$tag,$overlay,$needwrapper)=@_;
  $overlay=$overlay||T_OTHER("");
  ( T_CS('\group@item@maybe@unwrap'),
    ($needwrapper ? (Invocation(T_CS('\beamer@group@item@wrap'),$tag,$overlay)->unlist) : ()) )
DefConstructor('\beamer@group@item@wrap {} OptionalBeamerOverlay',
       "<omdoc:omtext ?#2(overlay='&overlayFrom(#2)')()>"
       . "?#1(<dc:title>#1</dc:title>)()"
             . "<omdoc:CMP>",
       beforeDigest=>sub {
 Let('\group@item@maybe@unwrap','\group@item@unwrap');
 #$_[0]->bgroup;
```

[6] EdNote: this is just to keep latexml quiet, no real functionality here.

[7] EdNote: Deyan: We reuse the CMP itemizations defined in the omdoc.cls.ltxml binding, adjusting the parameters to be overlay-sensitive

```
return; },
       properties=>sub{ RefStepItemCounter(); });


#DefConstructor('\beamer@itemize@item[] OptionalBeamerOverlay',
#       "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
#     . "?#1(<dc:title>#1</dc:title>)()",
#       properties=>sub{ RefStepItemCounter(); });
DefConstructor('\beamer@enumerate@item[] OptionalBeamerOverlay',
       "<omdoc:li ?#2(overlay='&overlayFrom(#2)')() >"
     . "?#1(<dc:title>#1</dc:title>)()",
       properties=>sub{ RefStepItemCounter(); });
DefConstructor('\beamer@description@item[] OptionalBeamerOverlay',
       "<omdoc:di ?#2(overlay='&overlayFrom(#2)')() >"
       . "?#1(<omdoc:dt>#1</omdoc:dt>)()<omdoc:dd>", # trust di and dt to autoclose
       properties=>sub{ RefStepItemCounter(); });
⟨/ltxml⟩#$
```

Now, some things that are imported from the `pgf` and `beamer` packages:

```
⟨*ltxml⟩
DefMacro('\putgraphicsat{}{}{}','\mygraphics[#2]{#3}');
DefMacro('\putat{}{}','#2');
⟨/ltxml⟩
```

4.6 Finale

Finally, we set the slide body font to the sans serife, and we terminate the LaTeXML bindings file with a success mark for perl.

```
⟨cls⟩\ifnotes\else\sf\fi
⟨ltxml⟩1;
```

References

[Koh10] Michael Kohlhase. *metakeys.sty: A generic framework for extensible Metadata in LaTeX*. Self-documenting LaTeX package. Comprehensive TeX Archive Network (CTAN), 2010. URL: http://www.ctan.org/tex-archive/macros/latex/contrib/stex/metakeys/metakeys.pdf.

[Ste] *Semantic Markup for LaTeX*. Project Homepage. URL: http://trac.kwarc.info/sTeX/ (visited on 02/22/2011).