Slides and Course Notes for Jacobs University*

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

July 20, 2010

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

3 The Implementation

3.1 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⟩
\newif\ifnotes\notesfalse
\newif\ifproblems\problemstrue
\DeclareOption{notes}{\notestrue}
\DeclareOption{slides}{\notesfalse}
\DeclareOption{noproblems}{\problemsfalse}
\ifnotes\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[notheorems,noamsthm]{beamerarticle}
```

```
\else
\LoadClass[notheorems,noamsthm,10pt]{beamer}
\newcounter{Item}
\newcounter{paragraph}
\newcounter{subparagraph}
\newcounter{Hfootnote}
\usetheme{Jacobs}
\fi
⟨/cls⟩
⟨*ltxml⟩
LoadClass('omdoc');
⟨/ltxml⟩
```

now, we load the remaining packages for both versions.

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

BegOP(1)

The `etex.sty` package is needed in the LaTeX, since it makes the extended the number of internal registers of the `etex` program available for processing. These tend to run out with sTeX otherwise.

EndOP(1)

3.2 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'));
```

[1] Old Part: we do not seem to load this any more, what to do there?

```
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.

`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}
\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' ?#locator(stex:srcref='#locator')()>"
  .  "#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 ?#locator(stex:srcref='#locator')()><dc:title>#1</dc:title></omdoc:metadat
⟨/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(2)

[2]

3.3 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}}
```

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

```
\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(3)

Now, we set up the slide label for the `article` mode[3]

`\slidelabel`

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

3.4 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\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.

```
⟨*cls⟩
\def\textwarning{\raisebox{-.05cm}{\includegraphics[width=1.2em]{dangerous-bend}}\xspace}
⟨/cls⟩
⟨*ltxml⟩
DefMacro('\textwarning',"");
⟨/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⟩
\newenvironment{ttitle}{\begin{center}\LARGE\begin{tabular}{|c|}\hline}%
```

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

```
 {\\\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⟩
DefEnvironment('{titleslide}',"");
DefEnvironment('{titleslide}',"<omdoc:omgroup ?#locator(stex:srcref='#locator')()>#body</omdoc:
DefEnvironment('{ttitle}', "\n<Title>#body</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⟩
```

3.5 Miscellaneous

Some shortcuts that use math symbols but are not mathematical at all; in particular, they should not be translated by LaTeXML.

```
⟨*cls⟩
\newcommand\hateq{\ensuremath{\hat=}}
\newcommand\textleadsto{\ensuremath{\leadsto}}
⟨/cls⟩
⟨*ltxml⟩
DefConstructor('\hateq',"=\x{03C2}");
DefConstructor('\textleadsto',"=\x{219D}");
⟨/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.[4]

EdNote(4)

[5]

EdNote(5)

```
⟨*ltxml⟩
DefConstructor('\uncover','#1');
```

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

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

```
#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,$overlay)->unlist) : ()) )
DefConstructor('\beamer@group@item@wrap {} OptionalBeamerOverlay',
       "<omdoc:omtext ?#locator(stex:srcref='#locator')() ?#2(overlay='&overlayFrom(#2)')()>"
       . "?#1(<dc:title>#1</dc:title>)()"
              . "<omdoc:CMP ?#locator(stex:srcref='#locator')()><omdoc:p ?#locator(stex:srcref
       beforeDigest=>sub {
 Let('\group@item@maybe@unwrap','\group@item@unwrap');
 #$_[0]->bgroup;
 useCMPItemizations();
 return; },
       properties=>sub{ RefStepItemCounter(); });


DefConstructor('\beamer@itemize@item[] OptionalBeamerOverlay',
       "<omdoc:li ?#locator(stex:srcref='#locator')() ?#2(overlay='&overlayFrom(#2)')() >"
     . "?#1(<dc:title ?#locator(stex:srcref='#locator')()>#1</dc:title>)()",
       properties=>sub{ RefStepItemCounter(); });
DefConstructor('\beamer@enumerate@item[] OptionalBeamerOverlay',
       "<omdoc:li ?#locator(stex:srcref='#locator')() ?#2(overlay='&overlayFrom(#2)')() >"
     . "?#1(<dc:title ?#locator(stex:srcref='#locator')()>#1</dc:title>)()",
       properties=>sub{ RefStepItemCounter(); });
DefConstructor('\beamer@description@item[] OptionalBeamerOverlay',
       "<omdoc:di ?#locator(stex:srcref='#locator')()  ?#2(overlay='&overlayFrom(#2)')() >"
       . "?#1(<omdoc:dt ?#locator(stex:srcref='#locator')()>#1</omdoc:dt>)()<omdoc:dd>", # trus
       properties=>sub{ RefStepItemCounter(); });

#We ALWAYS use the beamer itemizations for the slides:
```

```
Let('\CMP@itemize@item'=>'\beamer@itemize@item');
Let('\CMP@enumerate@item'=> '\beamer@enumerate@item');
Let('\CMP@description@item'=> '\beamer@description@item');
Let('\group@item' =>'\beamer@group@item');
Let('\itemize@item'=>'\beamer@group@item');
Let('\enumerate@item'=>'\beamer@group@item');
Let('\description@item'=>'\beamer@group@item');
Let('\only'=>'\beamer@group@item');
⟨/ltxml⟩#$
```

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

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

3.6 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;
```