`hwexam.sty/cls`: An Infrastructure for formatting Assignments and Exams[*]

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

February 18, 2014

Abstract

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

Contents

[*]Version v1.0 (last revised 2013/12/12)

1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [Koh13c]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

2 The User Interface

2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Koh13a] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Koh13b] on which it is based and passes them on to that. For the `extrefs` option see [Koh13d].

2.2 Assignments

`assignment` `number` `title` `type` `given` `due`

This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. "quiz", or "homework"), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the LaTeX source.

`\testspace` `\testnewpage` `\testemptypage`

`\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` `duration` `min` `reqpts`

Finally, the `\testheading` takes an optional keyword argument where the keys `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

formats to

Name: Matriculation Number:

320101 General Computer Science (Fall 2010)

February 18, 2014

You have one hour(sharp) for the test;
Write the solutions to the sheet.
The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.
You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

Different problems test different skills and knowledge, so do not get stuck on one problem.

	To be used for grading, do not write here								
prob.	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	4	4	6	6	4	4	2	30	
reached									

good luck

Example 1: A generated test heading.

2.4 Including Assignments

\includeassignment

The `\includeassignment` macro can be used to include an assignment from another file. It takes an optional KeyVal argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

number
title
type
given
due

2.5 Support for MathHub

Much of the sTeX content is hosed on MathHub (`http://MathHub.info`), a portal and archive for flexiformal mathematics. MathHub offers GIT repositories (public and private escrow) for mathematical documentation projects, online and offline authoring and document development infrastructure, and a rich, interactive reading interface. The `modules` package supports repository-sensitive operations on MathHub.

Note that MathHub has two-level repository names of the form ⟨*group*⟩/⟨*repo*⟩, where ⟨*group*⟩ is a MathHub-unique repository group and ⟨*repo*⟩ a repository name that is ⟨*group*⟩-unique. The file and directory structure of a repository is arbitrary – except that it starts with the directory `source` because they are Math Archives in the sense of [Hor+11]. But this structure can be hidden from the sTeX author with MathHub-enabled versions of the `modules` macros.

`\includemhassignment` The `\includemhassignment` macro is a variant of `\includeassignment` with repository support. Instead of writing

```
\defpath{MathHub}{/user/foo/lmh/MathHub}
\includeassignment[pts=7]{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
\includemhassignment[fooMH/bar]{baz/foobar}
```

If `baz/foobar` is the "current module", i.e. if we are on the MathHub path ...`MathHub/fooMH/bar`..., then stating the repository in the first optional argument is redundant, so we can just use

```
\includemhassignment{baz/foobar}
```

Of course, neither LaTeX nor LaTeXMLknow about the repositories when they are called from a file system, so we can use the `\mhcurrentrepos` macro from the `modules` package to tell them. But this is only needed to initialize the infrastructure in the driver file. In particular, we do not need to set it in in each module, since the `\importmhmodule` macro sets the current repository automatically.

Caveat if you want to use the MathHub support macros (let's call them mh-variants), then every time a module is imported or a document fragment is included from another repos, the mh-variant `\importmhmodule` must be used, so that the "current repository" is set accordingly. To be exact, we only need to use mh-variants, if the imported module or included document fragment use mh-variants.

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 sTeX TRAC [sTeX].

1. none reported yet.

4 Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

`hwexam.dtx` generates four files: `hwexam.cls` (all the code between ⟨*cls⟩ and ⟨/cls⟩), `hwexam.sty` (between ⟨*package⟩ and ⟨/package⟩) and their LaTeXML bindings (between ⟨*ltxml.cls⟩ and ⟨/ltxml.cls⟩ and ⟨*ltxml.sty⟩ and ⟨/ltxml.sty⟩ respetively). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

4.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
⟨*cls⟩
\DeclareOption{test}{\PassOptionsToPackage{\CurrentOption}{hwexam}}
\DeclareOption{multiple}{\PassOptionsToPackage{\CurrentOption}{hwexam}}
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\DeclareOption{extrefs}{\PassOptionsToPackage{\CurrentOption}{sref}}
\DeclareOption{notes}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{hints}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{solutions}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{pts}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{min}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{boxed}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption{extract}{\PassOptionsToPackage{\CurrentOption}{problem}}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{omdoc}}
\ProcessOptions
⟨/cls⟩
⟨*ltxml.cls⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
use LaTeXML::Util::Pathname;
use Cwd qw(cwd abs_path);
DeclareOption('test',,sub {PassOptions('hwexam','sty',ToString(Digest(T_CS('\CurrentOption'))))
DeclareOption('multiple',sub {PassOptions('hwexam','sty',ToString(Digest(T_CS('\CurrentOption')
DeclareOption('showmeta',sub {PassOptions('metakeys','sty',ToString(Digest(T_CS('\CurrentOption
DeclareOption('extrefs',sub {PassOptions('sref','sty',ToString(Digest(T_CS('\CurrentOption'))))
DeclareOption('notes',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption')))
DeclareOption('hints',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption')))
DeclareOption('solutions',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption
DeclareOption('pts',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption'))));
DeclareOption('min',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption'))));
DeclareOption('boxed',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption')))
DeclareOption('extract',sub {PassOptions('problem','sty',ToString(Digest(T_CS('\CurrentOption')
DeclareOption(undef,sub {PassOptions('omdoc','cls',ToString(Digest(T_CS('\CurrentOption')))); }
ProcessOptions();
⟨/ltxml.cls⟩
```

We load `article.cls`, and the desired packages. For the LaTeXML bindings, we make sure the right packages are loaded.

```
⟨*cls⟩
\LoadClass{omdoc}
\RequirePackage{stex}
\RequirePackage{hwexam}
\RequirePackage{graphicx}
\RequirePackage{a4wide}
\RequirePackage{amssymb}
\RequirePackage{amstext}
\RequirePackage{amsmath}
⟨/cls⟩
⟨*ltxml.cls⟩
LoadClass('omdoc');
RequirePackage('stex');
RequirePackage('hwexam');
RequirePackage('graphicx');
RequirePackage('amssymb');
RequirePackage('amstext');
RequirePackage('amsmath');
⟨/ltxml.cls⟩
```

5 Implementation: The hwexam Package

5.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. Some come with their own conditionals that are set by the options, the rest is just passed on to the `problems` package.

```
⟨*package⟩
\DeclareOption{showmeta}{\PassOptionsToPackage{\CurrentOption}{metakeys}}
\newif\iftest\testfalse
\newif\ifsolutions\solutionsfalse
\DeclareOption{test}{\testtrue\solutionsfalse}
\newif\ifmultiple\multiplefalse
\DeclareOption{multiple}{\multipletrue}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
\ProcessOptions
⟨/package⟩
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
⟨*package⟩
\RequirePackage{keyval}[1997/11/10]
\RequirePackage{problem}
⟨/package⟩
```

Here comes the equivalent header information for LaTeXML, we also initialize the package inclusions. Since LaTeXML does not handle options yet, we have nothing to do.

```
⟨*ltxml⟩
# -*- CPERL -*-
package LaTeXML::Package::Pool;
use strict;
use LaTeXML::Package;
RequirePackage('problem');
⟨/ltxml⟩
```

Then we register the namespace of the requirements ontology

```
⟨*ltxml⟩
RegisterNamespace('assig'=>"http://omdoc.org/ontology/assignments#");
RegisterDocumentNamespace('assig'=>"http://omdoc.org/ontology/assignments#");
⟨/ltxml⟩
```

5.2 Assignments

We will prepare the keyval support for the `assignment` environment.

```
⟨*package⟩
\srefaddidkey{assig}
\addmetakey{assig}{number}
\addmetakey*{assig}{title}
\addmetakey{assig}{type}
\addmetakey{assig}{given}
\addmetakey{assig}{due}
```

The next three macros are intermediate functions that handle the case gracefully, where the respective token registers are undefined.

The `\given@due` macro prints information about the given and due status of the assignment. Its arguments specify the brackets.

```
\def\given@due#1#2{%
\ifx\assig@given\@empty
\ifx\assig@due\@empty\else#1 Due \assig@due #2\fi%
\else%assig@given non-empty
#1Given \assig@given%
\ifx\assig@due\@empty\else, Due \assig@due\fi #2\fi}
```

With them, we can define the central `assignment` environment. This has two forms (separated by `\ifmultiple`) in one we make a title block for an assignment sheet, and in the other we make a section heading and add it to the table of contents.

`assignment@titleblock` This macro prints the title block of a section. If the `multiple` package option is given we make a section heading out of this, and if not, a title block. Note that as `problem`s are numbered by section, we also set the section counter in the latter case.

```
\ifmultiple
```

```
\def\assignment@titleblock{%
\ifx\assig@number\@empty\stepcounter{section}\else\setcounter{section}{\assig@number}\fi%
\section*{\protect\document@hwexamtype~\arabic{section}:~\assig@title%
\ifx\assig@title\@empty\else\strut\\\fi\given@due()}%
\addcontentsline{toc}{section}%
{\document@hwexamtype~{\arabic{section}}:~%
\string\importmodules{\imported@modules}\assig@title}%
\setcounter{problem}{0}}
\else% multiple
\def\assignment@titleblock{%
\ifx\assig@number\@empty\stepcounter{section}\else\setcounter{section}{\assig@number}\fi%
\begin{center}\bf
\Large\@title\strut\\
\document@hwexamtype~\assig@number:~\assig@title\strut\\
\large{\given@due()}
\end{center}}
\fi %multiple
```

`assignment@process@keys` this macro collects the keys from its argument and corrects them from the outside.

```
\def\assignment@process@keys#1{\metasetkeys{assig}{#1}
\ifx\inclassig@title\@empty\else\def\assig@title{\inclassig@title}\fi
\ifx\inclassig@type\@empty\else\def\assig@type{\inclassig@type}\fi
\ifx\inclassig@number\@empty\else\def\assig@number{\inclassig@number}\fi
\ifx\inclassig@due\@empty\else\def\assig@due{\inclassig@due}\fi
\ifx\inclassig@given\@empty\else\def\assig@given{\inclassig@given}\fi}
```

for this to work we need to define the `\inclassig` macros in case no `\includeassignment` is ever called.

```
\def\inclassig@title{}
\def\inclassig@type{}
\def\inclassig@number{}
\def\inclassig@due{}
\def\inclassig@given{}
```

`assignment`

```
\newenvironment{assignment}[1][]{\assignment@process@keys{#1}%
\assignment@titleblock%
\def\currentsectionlevel{assignment\xspace}%
\def\Currentsectionlevel{Assignment\xspace}%
\ignorespaces}{}
⟨/package⟩
```

```
⟨*ltxml⟩
DefEnvironment('{assignment} OptionalKeyVals:assig',
  "<omdoc:omgroup ?&GetKeyVal(#1,'id')(xml:id='&GetKeyVal(#1,'id')')() "
  .  "assig:dummy='for the namespace'>"
 .  "<omdoc:metadata>"
  .     "<dc:title>"
  .        "Assignment ?&GetKeyVal(#1,'num')(&GetKeyVal(#1,'num').)()"
  .        "?&GetKeyVal(#1,'title')((&GetKeyVal(#1,'title')))"
.    "</dc:title>"
.    "?&GetKeyVal(#1,'given')(<omdoc:meta property='assig:given'>&GetKeyVal(#1,'given')</omdo
.    "?&GetKeyVal(#1,'due')(<omdoc:meta property='assig:due'>&GetKeyVal(#1,'due')</omdoc:meta
.    "?&GetKeyVal(#1,'pts')(<omdoc:meta property='assig:pts'>&GetKeyVal(#1,'pts')</omdoc:meta
.  "</omdoc:metadata>"
.  "#body"
."</omdoc:omgroup>\n",
afterDigest=> sub {
  my ($stomach, $kv) = @_;
  my $kvi = LookupValue('inclassig');
  my @keys = qw(id num title pts given due);
  my @vals = $kvi && map($kvi->getValue($_), @keys);
  foreach my $i(0..$#vals) {
     $kv->setValue($keys[$i],$vals[$i]) if $vals[$i];
   }});#$
⟨/ltxml⟩
```

```
⟨*package⟩
\def\assig@default@type{Assignment}
\addmetakey[\assig@default@type]{document}{hwexamtype}
⟨/package⟩
```

5.3 Including Assignments

`\in*assignment` This macro is essentially a glorified `\include` statement, it just sets some internal macros first that overwrite the local points Importantly, it resets the `inclassig` keys after the input.

```
⟨*package⟩
\addmetakey{inclassig}{number}
\addmetakey{inclassig}{title}
\addmetakey{inclassig}{type}
\addmetakey{inclassig}{given}
\addmetakey{inclassig}{due}
\newcommand\includeassignment[2][]{\metasetkeys{inclassig}{#1}%
\include{#2}\clear@inclassig@keys}
\newcommand\inputassignment[2][]{\metasetkeys{inclassig}{#1}%
\input{#2}\clear@inclassig@keys}
⟨/package⟩
⟨*ltxml⟩
DefMacro('\includeassignment [] {}', sub {
  my ($stomach, $arg1, $arg2) = @_;
  AssignValue('inclassig',$arg1) if $arg1;
  (Invocation(T_CS('\input'),$arg2)->unlist);
});
DefMacro('\inputassignment [] {}','\includeassignment[#1]{#2}');
⟨/ltxml⟩
```

5.4 Typesetting Exams

```
⟨*package⟩
\addmetakey{quizheading}{tas}
\newcommand\quizheading[1]{\def\@tas{#1}%
\large\noindent NAME: \hspace{8cm}  MAILBOX:\\[2ex]%
\ifx\@tas\@empty\else%
\noindent TA: \@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\\[2ex]\fi}
```

```
\addmetakey{testheading}{min}
\addmetakey{testheading}{duration}
\addmetakey{testheading}{reqpts}
\newenvironment{testheading}[1][]{\metasetkeys{testheading}{#1}
{\noindent\large{}Name: \hfill Matriculation Number:\hspace*{2cm}\strut\\[1ex]
\begin{center}\Large\textbf{\@title}\\[1ex]\large\@date\\[3ex]\end{center}
{\textbf{You have
\ifx\test@heading@duration\@empty\testheading@min minutes\else\testheading@duration\fi
(sharp) for the test}};\\ Write the solutions to the sheet.}\par\noindent

\newcount\check@time\check@time=\testheading@min
\advance\check@time by -\theassignment@totalmin
The estimated time for solving this exam is {\theassignment@totalmin} minutes,
leaving you {\the\check@time} minutes for revising your exam.

\newcount\bonus@pts\bonus@pts=\theassignment@totalpts
\advance\bonus@pts by -\testheading@reqpts
You can reach {\theassignment@totalpts} points if you solve all problems. You will only need
{\testheading@reqpts} points for a perfect score, i.e.\ {\the\bonus@pts} points are
bonus points. \vfill
\begin{center}
  {\Large\em
%  You have ample time, so take it slow and avoid rushing to mistakes!\\[2ex]
  Different problems test different skills and knowledge, so do not get stuck on
  one problem.}\vfill\par\correction@table \\[3ex]
\end{center}}
{\newpage}
⟨/package⟩
⟨*ltxml⟩
DefEnvironment('{testheading}OptionalKeyVals:omdoc','');
⟨/ltxml⟩
```

```
⟨*package⟩
\def\testspace#1{\iftest\vspace*{#1}\fi}
\def\testnewpage{\iftest\newpage\fi}
\def\testemptypage{\iftest\begin{center}This page was intentionally left
    blank for extra space\end{center}\vfill\eject\else\fi}
⟨/package⟩
⟨*ltxml⟩
DefConstructor('\testspace{}','');
DefConstructor('\testnewpage','');
DefConstructor('\testemptypage','');
⟨/ltxml⟩
```

`\@problem` This macro acts on a problem's record in the `*.aux` file. Here we redefine it to generate the correction table.

```
⟨*package⟩
\def\@problem#1#2#3{\stepcounter{assignment@probs}
\def\@test{#2}\ifx\@test\@empty\else\addtocounter{assignment@totalpts}{#2}\fi
\def\@test{#3}\ifx\@test\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
\xdef\correction@probs{\correction@probs & #1}%
\xdef\correction@pts{\correction@pts & #2}
\xdef\correction@reached{\correction@reached &}}
⟨/package⟩
```

`\correction@table` This macro generates the correction table

```
⟨*package⟩
\newcounter{assignment@probs}
\newcounter{assignment@totalpts}
\newcounter{assignment@totalmin}
\def\correction@probs{prob.}%
\def\correction@pts{total}%
\def\correction@reached{reached}%
\stepcounter{assignment@probs}
\def\correction@table{\begin{tabular}{|l|*{\theassignment@probs}{c|}|p{3cm}|}\hline%
&\multicolumn{\theassignment@probs}{c||}%|
{\footnotesize To be used for grading, do not write here} &\\\hline
\correction@probs & Sum & grade\\\hline
\correction@pts &\theassignment@totalpts & \strut\hspace{3cm}\strut\\\hline
\correction@reached & & \\[.7cm]\hline
\end{tabular}}
⟨/package⟩
```

5.5 Support for **MathHub**

`\includemhassignment` The `\includemhassignment` saves the current value of `\mh@currentrepos` in a local macro `\mh@@repos`, resets `\mh@currentrepos` to the new value if one is given in the optional argument, and after importing resets `\mh@currentrepos` to the old value in `\mh@@repos`.

```
⟨ltxml⟩RawTeX('
⟨*ltxml | package⟩
\newcommand\includemhassignment[2][]{\metasetkeys{inclassig}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\inclassig@mhrepos\@empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
\includeassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos\clear@inclassig@keys}
```

`\inputmhassignment` analogous

```
\newcommand\inputmhassignment[2][]{\metasetkeys{inclassig}{#1}%
\edef\mh@@repos{\mh@currentrepos}%
\ifx\inclassig@mhrepos\@empty\else\mhcurrentrepos\inclassig@mhrepos\fi%
\inputassignment[#1]{\MathHub{\mh@currentrepos/source/#2}}%
\mhcurrentrepos\mh@@repos\clear@inclassig@keys}
⟨/ltxml | package⟩
⟨ltxml⟩');
```

5.6 Leftovers

at some point, we may want to reactivate the logos font, then we use

```
here we define the logos that characterize the assignment
\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\def\bierglas{{\bierfont\char65}}
\def\denker{{\denkerfont\char65}}
\def\uhr{{\uhrfont\char65}}
\def\warnschild{{\warnschildfont\char 65}}
\def\hardA{\warnschild}
\def\longA{\uhr}
\def\thinkA{\denker}
\def\discussA{\bierglas}
```

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

```
⟨ltxml⟩1;
```

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.