The moodle package: generating Moodle quizzes via LaTeX[*]

Anders Hendrickson
St. Norbert College, De Pere, WI, USA
anders.hendrickson@snc.edu

January 11, 2016

1 Motivation

The acronym Moodle stands for "Modular Object-Oriented Dynamic Learning Environment." It is an open source learning management system employed by many universities, colleges, and high schools to provide digital access to course materials, such as notes, video lectures, forums, and the like; see https://moodle.com/moodle-lms/ for more information. One of the many useful features of Moodle is that mathematical and scientific notation can be entered in LaTeX or TeX code, which will be typeset either through a built-in TeX filter or by invoking MathJax.

For instructors who want to give students frequent feedback, but lack the time to do so, a particularly valuable module in Moodle is the *quiz*. A Moodle quiz can consist of several different types of questions—not only multiple choice or true/false questions, but also questions requiring a short phrase or numerical answer, and even essay questions. All but the essay questions are automatically graded by the system, and the instructor has full control over how often the quiz may be attempted, its duration, and so forth. Feedback can be tailored to specific mistakes the student makes.

All these features make Moodle quizzes very useful tools for instructors who have access to them. Unfortunately, the primary way to create or edit a Moodle quiz is through a web-based interface that can be slow to operate. To users of LaTeX, accustomed to the speed of typing source code on a keyboard alone, the agonizing slowness of switching between mouse and keyboard to navigate a web form with its myriad dropdown boxes, radio buttons, compounded with a perceptible time lag as one's Moodle server responds to requests, can produce a very frustrating experience. Moreover, editing is entirely impossible without network access.

[*] This document corresponds to moodle.sty v0.5, dated 2016/01/11.

Once the quiz is written, there is no easy way to view and proofread all the information of which it is made. Each question is edited on a separate webpage, which is so full of options that it cannot be viewed on a single screen. An instructor has to spend much time checking over the newly created quiz in order to be confident there are no errors.

Added to all this is the frustration of managing graphics. If a question requires an image—say, asking a calculus student to interpret the graph of a function—the image must first be produced as a standalone file (e.g., in JPG or PNG format), uploaded to Moodle, and then chosen in a web-based HTML editor. Great is the vexation of the instructor who decided to alter a question, as there are more and more possibilities of error whenever multiple files must be kept synchronized.

Users of LaTeX are also accustomed to the speed and flexibility that comes from defining their own macros, which may be as brief as writing `\R` instead of `\mathbb{R}` or as complex as macros that generate entire paragraphs of text. The Moodle editor, by contrast, requires you to type `\mathbb{R}` every single time you want $\mathbb{R}$.

Finally, there is the question of archiving and reusing one's work. Much, much work goes into creating Moodle quizzes, which then reside on a Moodle server somewhere in the cloud in a format neither easily browsable nor easily modifiable.

LaTeX itself has the power to solve all these difficulties: it is swift to edit and swifter to compile a LaTeX document, and the PDF may be previewed onscreen or printed out for ease of proofreading. Mathematical graphics can be integrated within the main file through Ti*k*Z, and of course LaTeX macros can be customized. Using the present moodle.sty package, a quiz author can type a quiz using familiar LaTeX syntax and document structure. Upon compilation, LaTeX will generate both a well-organized PDF that is easy to proofread and an XML file that can be uploaded directly to Moodle. The entire process is far faster than using Moodle's own web-based editor, makes it easier to catch one's mistakes, and the ultimate source code of one's work is a human-readable `.tex` file that can be archived, browsed, and edited offline.

2 Workflow

The process of creating a quiz in Moodle using this package is as follows:

1. Write a LaTeX document using `\usepackage{moodle}` as described below.
2. Compile the document to PDF using pdfLaTeX. This will also produce the file ⟨*jobname*⟩`-moodle.xml`.
3. Open Moodle, navigate to the desired course, and under "Question bank" select "Import."
4. Select "Moodle XML format," choose the XML file to upload, and press "Import."
5. After Moodle verifies that the questions have been imported correctly, you may add them to your quizzes.

3 Usage

3.1 Example Document

The following pages presume the reader already has some familiarity with creating and editing Moodle quizzes through the web interface. The `xkeyval` package is used to provide a key-value interface. Here is a simple example document:

```
\documentclass[12pt]{article}
\usepackage{moodle}

\begin{document}

\begin{quiz}{My first quiz}
  \begin{numerical}[points=2]{Basic addition}
    What is $8+3$?
    \item 11
  \end{numerical}

  \begin{shortanswer}[case sensitive=true]{Newton's name}
    What was Newton's first name?
    \item Isaac
    \item[fraction=0, feedback={No, silly!}] Fig
    \item{fraction=0} Sir
  \end{shortanswer}

  \begin{multi}[points=3]{A first derivative}
    What is the first derivative of $x^3$?
    \item  $\frac{1}{4} x^4+C$
    \item* $3x^2$
    \item  $51$
  \end{multi}
\end{quiz}

\end{document}
```

Key features to note in this first example are that a `quiz` environment contains several question environments. Each question takes a name as a mandatory argument, and it may also take optional key-value arguments within brackets. The question environments resemble list environments such as `itemize` or `enumerate`, in that answers are set off by `\item`'s, but the question itself is the text that occurs before the first `\item`.

`draft` If the package option `draft` is invoked, by calling `\usepackage[draft]{moodle}`, then no XML file will be generated. This is especially useful while editing a quiz containing graphics, so as to avoid the time spent converting image files.

3.2 Quiz and Question Environments

`quiz` A `.tex` document to generate Moodle quizzes contains one or more `quiz` environments, within which various question environments are nested. The required argument to the `quiz` environment names the "question bank" to which the questions inside will belong after being uploaded to moodle.

```
\begin{quiz}[⟨options⟩]{⟨question bank name⟩}
```

There are no `quiz`-specific options, but any ⟨*options*⟩ set with `\begin{quiz}` will be inherited by all questions contained within that `quiz` environment.

`\moodleset` Options may also be set at any time with `\moodleset{`⟨*options*⟩`}`; these changes are local to TEX-groups.

The syntax for each question environment is

```
\begin{⟨question type⟩}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item ⟨item⟩
       ⋮
    \item ⟨item⟩
\end{⟨question type⟩}
```

The meaning of the ⟨*item*⟩s varies depending on the question type, but they usually are answers to the question. Details will be given below.

The following key-value options may be set for all questions:

`points` `default grade` By default, each question is worth 1 point on the quiz. This may be changed with the `points` key or its synonym, `default grade`; for example, `points=2` makes that question worth two points.

`penalty` The `penalty` is the fraction of points that is taken off for each wrong attempt; it may be set to any value between 0 and 1. The default is `penalty=0.10`.

`fraction` In most question types, it is possible to designate some answers as being worth partial credit—that is, some fraction of a completely correct answer. The `fraction` key may be set to any integer from `0` (entirely wrong) to `100` (entirely correct).

`feedback` The `feedback` key sets text that will appear to the student after completing the quiz. For example, one might set

```
feedback={A question like this will appear on the exam.}
```

The desired feedback should be included in braces.

Two kinds of feedback can be given. If the `feedback` key is set for a question, then that feedback will appear to each student regardless of the student's answer. Answer-specific feedback (perhaps explaining a common mistake) may also be given by setting the `feedback` key *at the individual answer.*

3.3 Question Types

We next discuss the various question types supported by moodle.sty and the options that may be set.

3.3.1 Multiple Choice

multi

The syntax for a classic multiple choice question, with only one correct answer, is as follows:

```
\begin{multi}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item* ⟨correct answer⟩
    \item[⟨options⟩] ⟨wrong answer⟩
      ⋮
    \item[⟨options⟩] ⟨wrong answer⟩
\end{multi}
```

The correct answer is designated by the asterisk * after the \item; it need not appear first in the list.

shuffle

The boolean key shuffle determines whether Moodle will rearrange the possible answers in a random order. Setting shuffle=false will guarantee that the answer appear in the order they were typed; the default is shuffle=true.

numbering

Moodle offers different options for numbering the possible answers. You may set the numbering key to any of the following values, which mirror the usual LaTeX syntax: alph, Alph, arabic, roman, Roman, and none. Calling numbering=none produces an unnumbered list of answers. The Moodle syntax of abc, ABC, 123, iii, and IIII is also acceptable, but note that it requires *four* I's to obtain upper-case Roman numerals this way.

fraction

The fraction key can be used to designate some wrong answers as being worth partial credit. For example, a question might read thus:

```
\begin{multi}{my question}
  Compute $\int 4x^3\,dx$.
  \item* $x^4+C$
  \item[fraction=50] $x^4$
  \item $12x^2$
\end{multi}
```

Thus the asterisk * is shorthand for fraction=100, whereas a bare \item sets fraction=0.

single
multiple

By default, the multi environment produces a multiple choice question operates with only one correct answer; this is called single mode, and on Moodle it appears with radio buttons. It is also possible to write questions with more than one correct answer, asking the user to check all correct answers. To do this, use the key multiple or single=false.

The worth of each correct answers in `multiple` mode may be set by `fraction`, but Moodle will require that all the fractions add up to at least 100. If you simply designate each correct answer with `\item*`, then moodle.sty will divide the 100% equally among those answers. For example, the following two examples produce equivalent results:

```
\begin{multi}[multiple]{my question}
  Which numbers are prime?
  \item* 5
  \item 6
  \item* 7
  \item 8
\end{multi}
```

```
\begin{multi}[multiple]{my question}
  Which numbers are prime?
  \item[fraction=50] 5
  \item 6
  \item[fraction=50] 7
  \item 8
\end{multi}
```

3.3.2 Numerical

A numerical question in Moodle requires the student to input a real number in decimal form. Its typical format is

```
\begin{numerical}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨correct answer⟩
\end{numerical}
```

If there is more than one correct answer, additional `\item`'s may be included. Because this is not a multiple choice question, there is no need to provide incorrect answers. There may nevertheless be reasons to include incorrect answers. For example, partially correct answers may be specified by setting the `fraction` key. Feedback for a common mistake may be given by including the incorrect answer like this:

```
\item[fraction=0,feedback={You forgot to antidifferentiate!}] ⟨incorrect answer⟩
```

3.3.3 Short Answer

A short answer question resembles a numerical question: the student is to fill in a text box with a missing word or phrase.

```
\begin{shortanswer}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨correct answer⟩
      ⋮
    \item[⟨options⟩] ⟨correct answer⟩
\end{shortanswer}
```

You can make the text box appear as part of the question with the control sequence `\blank`. For example, your question might read

```
\begin{shortanswer}{Leibniz}
  Newton's rival was Gottfried Wilhelm \blank.
  \item Leibniz
  \item Leibniz.
\end{shortanswer}
```

Note that as the blank occurred at the end of a sentence, we included two answers, lest students get the question wrong merely by including or omitting a period.

`case sensitive`
`usecase`

By default, Moodle ignores the distinction between upper- and lower-case letters when grading a short answer question. You can make a question case-sensitive with the key `case sensitive` or its shorter synonym `usecase`.

3.3.4 Essay Questions

Instructors may ask essay questions on a Moodle quiz, although Moodle's software is not up to the task of grading them! Instead each essay question answer must be graded manually by the instructor or a teaching assistant.

```
\begin{essay}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨notes for grader⟩
      ⋮
    \item[⟨options⟩] ⟨notes for grader⟩
\end{essay}
```

Instead of containing answers, the `\item` tags for the `essay` question contain notes that will appear to whoever is grading the question manually.

`response required`

Although Moodle cannot grade the content of an essay question, it can at least determine whether the question has been left blank. If the `response required` key is set, Moodle will insist that the student enter something in the blank before accepting the quiz as completed.

`response format` Moodle offers five different ways for students to enter and/or upload their answers to an essay question. You may choose one of these five options:

- `html` An editor with the ability to format HTML responses including markup for italics, boldface, etc. This is the default.
- `file` A file picker allowing the student to upload a file, such as a PDF or DOC file, containing the essay.
- `html+file` The same HTML editor as above, but with the ability to upload files as well. This permits some students to type answers directly into the web form, and others to compose their essays in another program first.
- `text` This editor allows only for entering plain text without any markup.
- `monospaced` This yields a plain text editor, without any markup, and with a fixed-width font. This could be useful for entering code snippets, for example.

`response field lines` The key `response field lines` controls the height of the input box. The default is `response field lines=15`.

`attachments allowed` The `attachments allowed` key controls *how many* attachments a student is allowed to upload. Permissible values are `0`, `1`, `2`, `3`, or `unlimited`.

`attachments required` You may also require the student to upload a certain number of attachments by setting `attachments required` to `0`, `1`, `2`, or `3`.

`response template` Finally, you may preload the essay question with a template that the student will edit and/or type over, with the key `response template={`⟨*template*⟩`}`. The ⟨*template*⟩ should be enclosed in braces.

3.4 Matching Questions

A matching question offers a series of subquestions and a set of possible answers from which to choose. If there are m questions and n possible answers, a matching question will look like this:

```
\begin{matching}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨question 1⟩ \answer ⟨answer 1⟩
    \item[⟨options⟩] ⟨question 2⟩ \answer ⟨answer 2⟩
      ⋮
    \item[⟨options⟩] ⟨question m⟩ \answer ⟨answer m⟩
    \item[⟨options⟩] \answer ⟨answer m + 1⟩
      ⋮
    \item[⟨options⟩] \answer ⟨answer n⟩
\end{matching}
```

Answers 1 through m correspond to questions 1 through m; answers $m+1$ through n are "decoy" answers. If multiple questions should have the same answer, be sure your typed answer match exactly, so that Moodle will not create duplicate copies of the same answer!

shuffle

The matching question accepts the option of shuffle to randomly permute the questions and answers; by default shuffle=true.

drag and drop
dd

Moodle has two formats for matching questions. The older matching question offered a dropdown box for choosing the answer to each question; the newer "drag and drop matching" format shows all questions in one column, all answers in a second column, and allows students to drag the correct answer to the question using a mouse. To enable drag-and-drop matching, use the key 'drag and drop' or 'dd' for short. The default is dd=false. If you choose the older format, then due to the limitations of dropdown boxes, no LaTeX or HTML code can be used in the answers.

3.5 Cloze Questions

A "cloze question" has one or more subquestions embedded within a passage of text. For example, you might ask students to fill in several missing words within a sentence, or calculate several coefficients of a polynomial. To encode cloze questions in LaTeX using this package is easy: you simply nest one or more multi, shortanswer, or numerical environments within a cloze environment, as in the following example:

```
\begin{cloze}{my cloze question} \\
  Thanks to calculus, invented by Isaac
  \begin{shortanswer}
    \item Newton
  \end{shortanswer},
  we know that the derivative of $x^2$ is
  \begin{multi}[horizontal]
    \item $2x$
    \item* $\frac{1}{3} x^3 + C$
    \item $0$
  \end{multi}
  and that $\int_0^2 x^2\,dx$ equals
  \begin{numerical}
    \item[tolerance=0.0004] 2.667
  \end{numerical}.
  Thanks, Isaac!
\end{cloze}
```

Note that when used as a subquestion within a cloze question, \begin{multi} is *not* followed by name in braces; the same is true for the shortanswer and numerical environments.

vertical
horizontal

Within a cloze question, by default a multiple choice question is implemented as an inline dropdown box. This is visually compact, but it also prevents the use of mathematical or HTML formatting. Adding the option vertical displays the subquestion as a vertical column of radio buttons instead; likewise the option horizontal creates a horizontal row of radio buttons.

4 Conversion to HTML

Questions should be typed as usual for LaTeX, including `\$` to obtain dollar signs, `$`'s for math shifts, `$$`'s for display math, et cetera. The package moodle.sty automatically converts this LaTeX code into HTML for web display.

The following table lists LaTeX commands and environments that will be converted to HTML:

```
~                     \#          \$
\&                    \\          \S
\{                    \}          \␣
\emph                 \textbf     \par
\ldots                \dots       \relax
\euro                 \texteuro
\begin{center}
\begin{enumerate}
\begin{itemize}
\begin{tikzpicture}
```

Single and double quotation marks and the diacritical commands `\^`, `\'`, ``\` ``, `\"`, `\~`, and `\H` are also handled, as are the characters `\aa`, `\ae`, `\o`, `\ss`, `\l`, and their capitalizations.

In addition, `<` and `>` will be converted to `&lt;` and `&gt;` *within math mode only.* If they should be typed outside of math mode, they will be passed as typed to the HTML, and probably interpreted by students' browsers as HTML tags or other unpredicated results.

Be aware that *moodle.sty does not know how to convert any other TeX or LaTeX commands to HTML.* If other sequences are used, they may be passed verbatim to the XML file or may cause unpredicted results.

If you think of another LaTeX command that should be changed to an HTML equivalent, please contact the author at `anders.hendrickson@snc.edu` so that it may be added to a future revision of the package.

5 Graphics

The moodle.sty package can handle two kinds of graphics seamlessly. External graphics files may be included with the `\includegraphics` command from the `graphicx` package, and graphics may be generated internally using TikZ. In either case, the graphics will be embedded in base-64 encoding directly within the Moodle XML produced. This prevents the hassle of managing separate graphics files on the Moodle server, as Moodle will store the picture within the question in the question bank.

`\includegraphics`
`height`
`width`

When using `\includegraphics`, the only options currently supported are `height` and `width`. Attempts to use other `\includegraphics` options, such as `scale` or `angle`, will affect the PDF but not the XML output. The dimensions set by `height` and `width` are TeX dimensions such as `4in` or `2.3cm`. In order to prepare the image for web viewing, this package converts those dimensions to pixels `ppi` using a default of 103 pixels per inch.[1] That value may be changed by setting the `ppi` key (e.g., `ppi=72`); this is probably best done for the entire document with a `\moodleset` command, rather than image-by-image.

The mechanism used is somewhat fragile and relies upon one Ti*k*Z library and three free external programs.

1. moodle.sty invokes `\usetikzlibrary{external}`, so that each `tikzpicture` environment is compiled to a freestanding PDF file.

2. GhostScript (`www.ghostscript.com`) is used to convert the PDF output from Ti*k*Z into a PNG raster graphics file. The default command line is presumed to be `gswin64c.exe` (if `\ifwindows` from the `ifplatform` package returns true) or `gs` (if `\ifwindows` returns false). If your system requires a different command line to invoke Ghostscript, you may change it by invoking `\ghostscriptcommand{`⟨*executable filename*⟩`}`.

   `\ghostscriptcommand`

3. When external graphics files such as JPG or GIF are included, the open-source ImageMagick software (`www.imagemagick.org`) converts each file to PNG format. The command line for ImageMagick is the nondescript word `convert`, but may be changed by invoking `\imagemagickcommand{`⟨*executable filename*⟩`}`.

   `\imagemagickcommand`

4. Finally, the OpenSSL toolkit (`www.openssl.org`) is used to base64-encode the PNG file so that it can be included directly in the text of the XML file. The command line is presumed to be `openssl`, but can be changed with `\opensslcommand{`⟨*executable filename*⟩`}`.

   `\opensslcommand`

Please note the following vital points to make the graphics handling work:

- As of now, graphics are only supported when compiling directly to a PDF with `pdflatex`. Including PS graphics or using Ti*k*Z with the DVI→PS workflow is not yet supported.

- Filenames should not contains spaces or special characters like `_` or `\`.

- You must have Ghostscript, ImageMagick, and OpenSSL installed on your system to fully use the graphics-handling capabilities of moodle.sty.

- LaTeX must be able to call system commands; that is, `\write18` must be enabled. For MikTeX, this means adding `--enable-write18` to the command line of `pdflatex`; for TeXLive, this means adding `--shell-escape=true`.

[1] This number was selected because an image with `<IMG HEIGHT=103 WIDTH=103 SRC="...">` showed up as almost exactly 1 inch tall and 1 inch wide on several of this author's devices and browsers as of January 2016.

6 Known Limitations and Call for Bug Reports

Some features of Moodle quizzes have not yet been implemented in moodle.sty. Here is a non-exhaustive list.

- Moodle's feature of designating feedback for correct, partially correct, and incorrect answers.
- Calculated questions; that is, automatically generated numerical questions using randomly chosen numbers.
- Hints
- Keywords
- So-called "description" questions.
- Shuffled answers in multiple choice cloze subquestions.

I have used Version 0.5 for one semester's teaching, but if other users adopt this package, I fully expect them to find bugs. *Please* send all bugs you find to `anders.hendrickson@snc.edu`, so that I can fix them for subsequent versions.

7 Compatibility

This package has been written for and tested with the implementation of Moodle 2.9 run by Moodlerooms for St. Norbert College in January 2016. Future versions of this package will probably include some support for specifying your version of Moodle in the `.tex` file to help ensure compatability.

As the ultimate purpose of this package is the generation of XML files, future versions of moodle.sty will attempt to maintain backwards compatability with earlier versions of regarding the XML output, apart from bug fixes. Backwards compatibility of the PDF output is not yet guaranteed, however, in case the author or users discover better ways for the PDF to display the underlying XML data to be proofread.

In other words, compiling your current `.tex` file with a future version of moodle.sty should produce the same XML file it does now (apart from bug fixes), but it might produce a more informative, and hence different, PDF output.

8 Unrelated Tip: Quality of Moodle TEX Images

This has nothing to do with `moodle.sty`, but is a Frequently Asked Question in is own right. On some servers, at least, Moodle's default "TEX Filter" for displaying mathematical notation is of abysmally poor quality, rending mathematics as low-resolution PNG's. One solution that has worked for me is to go to "Course Administration → Filters," turn "TEX Notation" *off*, but turn "MathJax" *on*. This forces TEX code to be rendered by MathJax instead of Moodle, producing much higher-quality results.