Examination

v0.0.1

https://github.com/SillyFreak/typst-packages/examination

Clemens Koza

Abstract

Examination is a library for building exams, tests, etc. with Typst. It provides utilities for common question types and supports creating grading keys and sample solutions.

Contents

I Introduction

Examination has three general areas of focus:

- It provides a selection of question writing utilities, such as multiple choice or true/false questions.
- It helps with grading information: record the points that can be reached for each question and make them available for creating grading keys.
- It supports the creation of sample solutions by allowing to switch between the normal and "pre-filled" exam.

Right now, providing a styled template is not part of this package's scope.

II Questions and question metadata

Let's start with a really basic example that doesn't really show any of the benefits of this library yet:

```
#import "../src/lib.typ": grading, question

// you usually want to alias this, as you'll need it often
#let q = question.q

#q(points: 2)[
  == Question

  #lorem(20)
]
```

Question
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

After importing the library's modules and aliasing an important function, we simply get the same output as if we didn't do anything. The one peculiar thing here is `points: 2`: this adds some metadata to the question. Right now, the following metadata fields are available:

- `category`: an arbitrary category identifier (string) that can be used to group questions during further processing
- `points`: a number specifying how many points can be reached in that question
- additionally `body`: the complete content that was rendered as the question

The two former fields are not used unless you explicitly do so; let's look at how to do that. Let's say we want to show the points in each question's header:

```
#show heading: it => {
  // here, we need to access the current question's metadata
  question.current(q => [#it.body #h(1fr) / #q.points])
}
```

Question / 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Here we're using the `question.current()` function to access the metadata of the current question. Like Typst's `locate()` function, ordinarily, any computation has to happen inside as it can only return content – however, see the function's documentation for an escape hatch.

III Grading

The final puzzle piece is grading. There are many different possibilities to grade a test; right now Examination does not try to depend on a specific convention, but it does assume that in the end, a test has a final score and that the grade is derived by looking how high that score is. If your test does not fit that schema, you can simply use less of the related features.

The first step in creating a grading scheme is determining how many points can be achieved in total, using `grading.total-points()`. Let's at the same time also look at question categories as a way to get subtotals:

```
#question.all(qs => [
  #let total = grading.total-points(qs)

  Total points: #total

  Points from hard questions:
  #grading.total-points(qs, filter: q => q.category == "hard")
])

#q(category: "hard", points: 6)[
  == Hard Question

  #lorem(20)
]
```

Total points: 8

Points from hard questions: 12

Hard Question / 6

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Question / 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Once we have the total points of the text figured out, we need to define the grading key. Let's say the grades are in a three-grade system of "bad", "okay", and "good". We could define these grades like this:

```
#question.all(qs => [
  #let total = grading.total-points(qs)

  #let grades = grading.grades([bad], total * 2/4, [okay], total * 3/4,

  #grades
])
```

```
(
  (body: [bad], lower-limit: none, upper-limit: 4),
  (body: [okay], lower-limit: 4, upper-limit: 6),
  (body: [good], lower-limit: 6, upper-limit: none),
)
```

Hard Question / 6

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Question / 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat.

Obviously we would not want to render this representation as-is, but `grading.grades()` gives us a convenient way to have all the necessary information, without assuming things like inclusive or exclusive point ranges. The `test.typ` example in the gallery has a more complete demonstration of a grading key.

IV Question templates

TODO

V Sample solutions

TODO

VI Module reference

VI.a `examination.question`

- q()
- current()

q(body: `content`, category: `string`, points: `integer`) -> `content`

Adds a question with its metadata, and renders it. The questions can later be accessed using the other functions in this module.

Parameters:

`body` (`content`) – the content to be displayed for this question

`category` (`string` = `none`) – an optional category to be added to the question's metadata

`points` (`integer` = `none`) – an optional point score to be added to the question's metadata

current(func: `function`, loc: `location`) -> `content | any`

Locates the most recently defined question; within a q() call, that is the question *currently* being defined. The located question's metadata is used to call the provided function.

If the optional `loc` is provided, that location is used and the function's result is returned directly; otherwise, the result will be converted to a `content` by the necessary call to `locate`.

Parameters:

`func` (`function`) – a function

`loc` (`location` = `none`) – a

counter `counter`

The question counter

VI.b `examination.grading`

- total-points()
- grades()

total-points(questions, filter)

grades(..args)