XSIM

v0.1pre-5 2017/03/01

eXercise Sheets IMproved

Clemens Niederberger

https://github.com/cgnieder/xsim

contact@mychemistry.eu

Table of Contents

1. Licence, Requirements and README

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

XSIM loads the packages expl3 [L3Pa], xparse [L3Pb], etoolbox [Leh15], booktabs [Fea05] and translations [Nie15]. All of these packages are present on a modern and up to date TeX distribution such as TeX Live or MiKTeX so no further action should be needed.

2. Motivation and Background

It has been quite a while since I first published exsheets [Nie17] in Juni 2012. Since then it has gained a user base and a little bit of popularity as the number of questions on tex.sx shows (98 at the time of writing). User questions, bug reports and feature requests improved it over the time. It still has a version number starting with a zero, though, which in my versioning system means I still consider it experimental.

This is due to several facts. It lacks a few features which I consider essential for a full version 1. For one thing it is not possible to have several kinds of exercises numbered independently. Using verbatim material such as listings inside exercises and solutions is not possible and the current workaround isn't that ideal either. One request which dates back quite a while now was to have different types of points to exercises…

All of those aren't easy to add due to the way exsheets is implemented right now. As a consequence I wanted to re-implement exsheets for a long time. This is what lead to XSIM. Internally the package works completely different. It will be the official successor of exsheets which is now considered obsolete and will only receive bugfix releases any more.

3. Package Options

XSIM has two package options:

`verbose`
Writes extensive information about what XSIM is doing into the log file.

`final`
If used the exercise and solution environments will not rewrite the environment body files.

Those options are used the usual way as package option

```
\usepackage[verbose]{xsim}
```

or as global option

```
\documentclass[verbose]{article}
```

or via the setup command:

`\xsimsetup{`⟨*options*⟩`}`
Set up XSIM's two package options and all other options described at other places in the manual.

4. How to Read the Manual

4.1. Nomenclature

Throughout this manual certain terms are used. This section explains their meaning in this manual.

collection A *collection* bundles a number of exercises of one type or all types of exercises within certain barriers in the document. Those exercise collections can be printed at any place in the document *after* the collection is complete.

goal *Goals* are a certain type of properties with a numerical value the sum of which is available throughout the document.

parameter *Parameters* are options of exercise types which are the same for each exercise of a type and can be retrieved and used in exercise templates.

property *Properties* are options of exercises which are individual for each exercise and can be retrieved and used in exercise templates.

tag *Tags* are a certain type of properties with a csv list as value which can be used for selective usage of exercises.

template *Templates* are generic code frameworks which are used for typesetting XSIM's objects such as exercises, solutions, or grading tables.

4.2. Setting Options

Apart from the package options already described in section 3 on the preceding page XSIM has further options. Those can be "toplevel" options or options belonging to a module.

`toplevel` = {⟨*value*⟩}
A toplevel option.

`module` » `sublebel` = {⟨*value*⟩} A sublevel option belonging to the module `module`

Both kinds of options are set with `\xsimsetup`:

```
\xsimsetup{
  toplevel = {value} ,
  module/sublevel = {value}
}
```

4.3. Command descriptions

Some commands do have a * symbol printed next to their names. This indicates that the command is expandable, *i. e.*, it is usable in an `\edef` or `\write` context and will expand according to its description. All other commands are engine protected, *i. e.*, in the sense of ε-TEX's `\protected`.

Some command name descriptions end with `\TF`.

`\SomeCommandTF`⟨*arguments*⟩{⟨*true*⟩}{⟨*false*⟩}
A command with maybe some arguments and ending with the two arguments ⟨*true*⟩ and ⟨*false*⟩.

This means two things: the command is a conditional which tests something and depending on the outcome of the test leaves either the ⟨*true*⟩ argument (`T`) or the ⟨*false*⟩ argument (`F`) in the input stream. It also means to additional commands exist:

`\SomeCommandT`⟨*arguments*⟩{⟨*true*⟩}
The same as `\SomeCommandTF` but only with the ⟨*true*⟩ argument and no ⟨*false*⟩ argument.

`\SomeCommandF`⟨*arguments*⟩{⟨*false*⟩}
The same as `\SomeCommandTF` but only with the ⟨*false*⟩ argument and no ⟨*true*⟩ argument.

5. Exercises and Solutions

5.1. The Environments

`\begin{exercise}`[⟨*properties*⟩]
Input and typeset an exercise. See section 8 on page 8 for details on exercise properties.

`\begin{solution}`[⟨*options*⟩]
Input and typeset the solution to the exercise of the previous `exercise` environment. See section 11 on page 15 for details on options of solutions.

```
\begin{exercise}
  A first example for an exercise.
\end{exercise}
\begin{solution}
```

```
    A first example for a solution.
\end{solution}
```

Exercise 1

A first example for an exercise.

As can be seen in the example a solution is not printed with the default setup. This can be changed using the following option.

`solution` » `print` = `true`|`false` Default: `false`

Set if solutions are printed or or not.

The option (belonging to the module `solution`) can either be set locally as option to the `solution` environment

```
\begin{solution}[print=true]
  A first example for a solution.
\end{solution}
```

or with the setup command for all following solutions:

```
\xsimsetup{
  solution/print = true
}
```

There is an completely analoguous option for the `exercise` environment:

`exercise` » `print` = `true`|`false` Default: `true`

Set if exercises are printed or or not.

See also section 9 on page 12.

6. How the Exercise Environments Work

Both enviroments write the contents of their bodies verbatim to external files following a certain naming structure:

- ⟨*jobname*⟩`-`⟨*type*⟩`-`⟨*id*⟩`-exercise|solution-body.tex`

The name starts with the name of the job (which is the name of the document itself) followed by type and id of the corresponding exercise and then followed by the environment type. For example both enviroments from the first example have been written to files named

- `xsim_manual-exercise-1-exercise-body.tex` and

- `xsim_manual-exercise-1-solution-body.tex`, respectively.

Details on the ⟨*type*⟩ of an exercise will be given in section 7. *The ⟨id⟩ of an exercise is a positive integer unique to each exercise environment regardless if the exercise is being printed or used at all.*

These external files are input when the respective exercise or solution is printed. An advantage of using external files is that *verbatim material is allowed* inside the environments. Each of those files contains some information about itself and where and why it was generated[1]:

```
%-------------------------------------------------------------------------
% file `xsim_manual-exercise-1-exercise-body.tex'
%   in folder `exercises/'
%
%     exercise of type `exercise' with id `1'
%
% generated by the `exercise' environment of the
%   `xsim' package v0.1pre-5 (2017/03/01)
% from source `xsim_manual' on 2017/03/02 on line 1
% -------------------------------------------------------------------------
    A first example for an exercise.
```

Arguably one downside of the approach using external files for each exercise and its solution is that your project folder will be cluttered with files. In order to deal with this somehow XSIM offers the following option:

`path` = {⟨*path name*⟩} (initially empty)
With this option a subfolder or path within the main project folder can be given. Exercises will be written to and included from this path. *The path must exist on your system before you can use it!* This document uses `path = {exercises}`.

7. New Exercise Types

It is easy to define new exercise environments together with a corresponding solution environment using the following command:

`\DeclareExerciseType{`⟨*type*⟩`}{`⟨*parameters*⟩`}`
Declare a new exercise type analoguous to the `exercise` and `solution` environments.

The existing environment pair has been defined as follows:

1. In this example the sourcecode line number is misleading as the example where the file was generated itself was an external file where the `exercise` environment indeed *was* on line 1.

```
\DeclareExerciseType{exercise}{
  exercise-env      = exercise ,
  solution-env      = solution ,
  exercise-name     = \XSIMtranslate{exercise} ,
  solution-name     = \XSIMtranslate{solution} ,
  exercise-template = subsection* ,
  solution-template = subsection*
}
```

The above already is an example for almost all parameters that can (and often must) be set. Here is the complete list:

`exercise-env` = {⟨*exercise environment name*⟩}
The name for the environment used for the exercises of type ⟨*type*⟩. *This parameter must be set.*

`solution-env` = {⟨*solution environment name*⟩}
The name for the environment used for the solutions of type ⟨*type*⟩. *This parameter must be set.*

`exercise-name` = {⟨*exercise name*⟩}
The name of the exercises of type ⟨*type*⟩ – used for typesetting. *This parameter must be set.*

`solution-name` = {⟨*solution name*⟩}
The name of the solutions of type ⟨*type*⟩ – used for typesetting. *This parameter must be set.*

`exercise-template` = {⟨*exercise template*⟩}
The template used for typesetting the exercises of type ⟨*type*⟩. *This parameter must be set.* See section 13 on page 16 for details on templates.

`solution-template` = {⟨*solution template*⟩}
The template used for typesetting the exercises of type ⟨*type*⟩. *This parameter must be set.* See section 13 on page 16 for details on templates.

`counter` = {⟨*counter name*⟩}
The counter used for the exercises of type ⟨*type*⟩. If not explicitly set the counter with the same name as `exercise-env` is used. Otherwise the specified counter is used. This enables to have different types of exercises sharing a common counter.

`counter` = {⟨*integer*⟩}
An internal parameter that is used to keep track of the number of exercises of a type.

It is possible to change some of the parameters after an exercise type has been defined. Those include `exercise-name`, `solution-name`, `exercise-template`, `solution-template`, and `counter`:

`\SetExerciseParameter`{⟨*type*⟩}{⟨*parameter*⟩}{⟨*value*⟩}
Usable to set a single parameter to a new value.

`\SetExerciseParameters{`⟨*type*⟩`}{`⟨*parameters*⟩`}`
Set several parameters at once. ⟨*parameters*⟩ is a csv list of key/value pairs.

If you try to set an already set but fixed parameter like `exercise-env` a warning will be written to the log file. If you set `counter` to another value you must make sure that the new value is a valid and defined counter.

8. Exercise Properties

8.1. Predefined Properties

Exercise like the `exercise` environment and possibly others defined with `\DeclareExerciseType` have a number of predefined properties:

`id` = `{`⟨*integer*⟩`}`
Holds the internal id of an exercise. *Cannot be set by the user.*

`ID` = `{`⟨*text*⟩`}`
Holds the user id of an exercise if defined. Otherwise it is equal to `id`.

`counter` = `{`⟨*integer*⟩`}`
Holds the counter value of an exercise. *Cannot be set by the user.*

`subtitle` = `{`⟨*text*⟩`}`
Holds the subtitle of an exercise.

`points` = `{`⟨*number*⟩`}`
Holds the reachable points of an exercise.

`bonus-points` = `{`⟨*number*⟩`}`
Holds the reachable bonus-points of an exercise.

`print` = `true`|`false`
Holds the print boolean of an exercise.

`use` = `true`|`false`
Holds the usage boolean of an exercise.

`tags` = `{`⟨*csv list of tags*⟩`}`
Holds the list of tags the exercise should be associated with.

`topics` = `{`⟨*csv list of topics*⟩`}`
Holds the list of topics the exercise should be associated with.

Some of these properties are fixed and cannot be set by the user. Those include `id` and `counter`. The others can be set using the optional argument of the exercise environment.

```
\begin{exercise}[subtitle={This is a subtitle}]
  An exercise where some properties have been set.
\end{exercise}
```

Exercise 2 *This is a subtitle*

An exercise where some properties have been set.

8.2. Declaring Own Properties

XSIM offers the possibility to declare additional exercise properties:

`\DeclareExerciseProperty*{`⟨*property*⟩`}`
Declares the property ⟨*property*⟩. If used with the optional star a unique property is defined which means that each exercise must have a property value distinct from all other exercises.

`\DeclareExercisePropertyAlias{`⟨*property 1*⟩`}{`⟨*property 2*⟩`}`
Declares ⟨*property 1*⟩ to be an alias of ⟨*property 2*⟩. This means that each time ⟨*property 2*⟩ is set ⟨*property 1*⟩ will be set to the same value *unless* it has been set already. As an example: property `ID` is an alias of property `id`.

This is better demonstrated with an example:

```
\begin{exercise}
  \lipsum[4] % from package `lipsum'
  \verb+\GetExerciseProperty{id}+: \GetExerciseProperty{id} \par
  \verb+\GetExerciseAliasProperty{ID}+: \GetExerciseAliasProperty{ID} \par
  \verb+\GetExerciseProperty{ID}+: \GetExerciseProperty{ID}
\end{exercise}
\begin{exercise}[ID=foo-bar]
  \lipsum[4]
  \verb+\GetExerciseProperty{id}+: \GetExerciseProperty{id} \par
  \verb+\GetExerciseAliasProperty{ID}+: \GetExerciseAliasProperty{ID} \par
  \verb+\GetExerciseProperty{ID}+: \GetExerciseProperty{ID}
\end{exercise}
```

Exercise 3

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

`\GetExerciseProperty{id}`: 3
`\GetExerciseAliasProperty{ID}`: 3
`\GetExerciseProperty{ID}`: 3

Exercise 4

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
`\GetExerciseProperty{id}`: 4
`\GetExerciseAliasProperty{ID}`: 4
`\GetExerciseProperty{ID}`: foo-bar

The power of properties will get more clear when reading section 13 on page 16 about templates.

8.3. A Special Kind of Property: Exercise Goals

Exercise goals are a generic concept in XSIM for exercise properties like `points` or `bonus-points`. Those are properties which can (only) get a decimal number as value the sum of which is calculated and available (after a compilation) throughout the document.

`\DeclareExerciseGoal{`⟨*goal*⟩`}`
Declare a new exercise goal named ⟨*goal*⟩ and also a property called ⟨*goal*⟩.

`\TotalExerciseTypeGoal{`⟨*type*⟩`}{`⟨*goal*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal ⟨*goal*⟩ for all exercises of type ⟨*type*⟩. ⟨*singular*⟩ andf ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\TotalExerciseGoal{`⟨*goal*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal ⟨*goal*⟩ for all exercises. ⟨*singular*⟩ andf ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\printpoints{`⟨*type*⟩`}`
Print the sum of points for all exercises of type ⟨*type*⟩ followed by an appropriate translation of the words “point” or “points”, respectively.[2] Defined in terms of `\TotalExerciseTypeGoal`.

`\printtotalpoints`
Print the sum of points for all exercises followed by an appropriate translation of the words “point” or “points”, respectivaly. Defined in terms of `\TotalExerciseGoal`.

2. See section 14 on page 20 for details on the definition and usage of language dependent words.

`\printbonus{`⟨*type*⟩`}`
Print the sum of bonus points for all exercises of type ⟨*type*⟩ followed by an appropriate translation of the words “point” or “points”, respectivaly. Defined in terms of `\TotalExerciseTypeGoal`.

`\printtotalbonus`
Print the sum of bonus points for all exercises followed by an appropriate translation of the words “point” or “points”, respectivaly. Defined in terms of `\TotalExerciseGoal`.

The two existing goals are defined with

```
\DeclareExerciseGoal{points}
\DeclareExerciseGoal{bonus-points}
```

8.4. A Special Kind of Property: Exercise Tags

Exercise tags are a generic concept in XSIM for exercise properties like `tags` or `topics`. Those are properties which can (only) get a csv list of strings as value. Those strings can be used to selectively use exercises. See section 9 on the next page for details on *usage* of exercises and the difference to *printing* an exercise and how to use exercise tags for selection.

`\DeclareExerciseTagging`
tag This defines an exercise tagging group name ⟨*tag*⟩. It also defines a property named ⟨*tag*⟩. In addition to options are defines: and option named ⟨*tag*⟩ which can be used for selection and an boolean option ⟨*tag*⟩`/ignore-untagged`.

The two existing tagging groups have been defined and preset with the following code:

```
\DeclareExerciseTagging{tags}
\DeclareExerciseTagging{topics}
\xsimsetup{tags/ingore-untagged=false}
```

this means that these options are available:

`tags` = {⟨*csv list of tags*⟩}
Choose the set of tags whose associated exercises should be printed.

`topics` = {⟨*csv list of topics*⟩}
Choose the set of tags whose associated exercises should be printed.

`tags` » `ignore-tagging` = `true`|`false` Default: `false`
If set to true exercises with no tags will be printed even if tags have been chosen with the option `tags`.

`topics` » `ignore-tagging` = `true`|`false` Default: `true`
If set to true exercises with no topics will be printed even if tags have been chosen with the option `tags`.

9. Using and Printing an Exercise

9.1. What the Environments do

When an exercise is started with `\begin{exercise}` (or other environments defined through `\DeclareExerciseType`) then different things happen depending on different settings:

- If the *insert mode* is active nothing happens, see section 10 on page 14 for details on this.
- Else the id integer is incremented.
- If the exercise is *used* the corresponding counter is stepped and the exercise is added to the "use list". The properties `counter` and `use` are updated accordingly.
- If an exercise is printed then it is also used. An exercise that isn't used cannot be printed. Being printed means two things: being added to the "print list" and being typeset at the position where the exercise is placed in the source file. If an exercise is *not printed but used* it means that the counter will be stepped. This can be useful for creating an exercise sheet only containing the solutions for some exercises.
- If an exercise is printed certain hooks and template code is inserted around the environment body.

```
\begin{exercise}[print=false]
  This exercise will not be printed but the exercise counter will be
  incremented nonetheless. Its solution will be printed in the list of
  solutions.
\end{exercise}
\begin{solution}
  The solution of the exercise that has not been printed.
\end{solution}
```

The schematic structure of an exercise is shown in figure 1 on the next page.

9.2. Environment Options & Hooks

For each exercise type there are the following options for both environments, the environments' names are the module names for the options (here using the "exercise" type):

`exercise » print = true|false` Default: `true`

Determines if exercises of type "exercise" are printed.

`exercise » use = true|false` Default: `true`

Determines if exercises of type "exercise" are used.

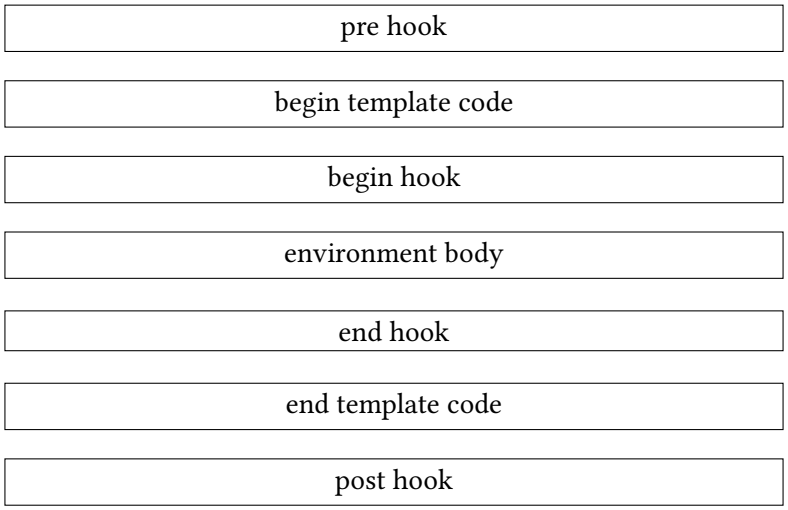

FIGURE 1: Schematic structure of an exercise or solution.

`exercise` » `pre-hook` = {⟨*code*⟩} (initially empty)
The code for the *pre exercise hook* for exercises of the type "exercise".

`exercise` » `begin-hook` = {⟨*code*⟩} (initially empty)
The code for the *begin exercise hook* for exercises of the type "exercise".

`exercise` » `end-hook` = {⟨*code*⟩} (initially empty)
The code for the *end exercise hook* for exercises of the type "exercise".

`exercise` » `post-hook` = {⟨*code*⟩} (initially empty)
The code for the *post exercise hook* for exercises of the type "exercise".

`solution` » `print` = `true`|`false` Default: `false`
Determines if solutions of type "exercise" are printed.

`solution` » `pre-hook` = {⟨*code*⟩} (initially empty)
The code for the *pre solution hook* for solutions of the type "exercise".

`solution` » `begin-hook` = {⟨*code*⟩} (initially empty)
The code for the *begin solution hook* for solutions of the type "exercise".

`solution` » `end-hook` = {⟨*code*⟩} (initially empty)
The code for the *end solution hook* for solutions of the type "exercise".

`solution` » `post-hook` = {⟨*code*⟩} (initially empty)
The code for the *post solution hook* for solutions of the type "exercise".

10. Collecting Exercises

XSIM knows the concept of "exercise collections". A collection must be declared in the preamble. Using a pair of commands explained below exercises between those commands are added to the corresponding collection but not printed. After a collection is completed the collection can be printed as often as needed.

`\DeclareExerciseCollection{`⟨*collection name*⟩`}`
Define a new collection ⟨*collection name*⟩ in the document preamble.

`\collectexercisestype{`⟨*collection name*⟩`}{`⟨*exercise type*⟩`}`
Opens the collection ⟨*collection name*⟩ which now collects all exercises of type ⟨*exercise type*⟩ until the collection is closed with `\collectexercisesstop`. Collections of other types are not collected.

`\collectexercises{`⟨*collection name*⟩`}`
Opens the collection ⟨*collection name*⟩ which now collects all exercises until the collection is closed with `\collectexercisesstop`.

`\collectexercisesstop{`⟨*collection name*⟩`}`
Closes the collection ⟨*collection name*⟩.

`\printcollection{`⟨*collection name*⟩`}`
Prints the collection ⟨*collection name*⟩, *i. e.*, all exercises collected earlier. This command cannot be used before the corresponding collection has been closed correctly.

The usage should be clear:

```
\collectexercises{foo}
\begin{exercise}
  This exercise is added to the collection `foo'.
\end{exercise}
\collectexercisesstop{foo}
```

Once the collection is closed it can be printed:

```
\printcollection{foo}
```

Exercise 6

This exercise is added to the collection 'foo'.

Actually a collection can be printed *before* it is opened, too. This needs at least two compilations, though.

You can open several collections at the same time:

```
\collectexercises{foo}
  ...
\collectexercisestype{bar}{exercises}
  ...
\collectexercisesstop{bar}
  ...
\collectexercisesstop{foo}
```

Exercises will be added to each open collection.

There is one generic collection called "`all exercises`". As the name already suggests it will hold all exercises. So if you say

```
\printcollection{all exercises}
```

all exercises will be printed.

If you use `\label`s inside of exercises and you print exercises more than once in your document (by reusing a collection for example) you will get

```
LaTeX Warning: There were multiply-defined labels.
```

Equally if you have environments like `\begin{exercise}` which step a counter the counter will be stepped each time the exercise is used.

11. Printing Solutions

There are two commands for printing the solutions to exercises – one for printing the solutions of a specific exercise type and another for printing all solutions.

`\printsolutionstype*[`⟨*options*⟩`]{`⟨*exercise type*⟩`}`
Prints the solutions of all used exercises of type ⟨*exercise type*⟩. The starred version only prints the solutions of all printed exercises of type ⟨*exercise type*⟩.

`\printsolutions*[`⟨*options*⟩`]`
Prints the solutions of all used exercises of all types. The starred version only prints the solutions of all printed exercises of all types, ordered by type.

```
\printsolutions
```

Solutions to the Exercises

Solution 1

A first example for a solution.

Solution 5

The solution of the exercise that has not been printed.

12. Grading Tables

13. Styling the Exercises – Templates

13.1. Background

Whenever XSIM outputs something to be typeset it uses so-called templates for the task. XSIM knows of three different kinds of templates:

- environment templates (see section 13.3.1),
- heading templates (see section 13.3.2) and
- grading table templates (see section 13.3.3)

The most important one for the styling of the exercises are the environment templates. Those templates give you complete control over the look and arrangement of an exercise. To be able to do this XSIM provides a large number of commands which can be used only inside template definitions.[3] Those commands are explained in the next section. Their usage will hopefully become clear in the examples in section 13.4.

13.2. Commands for Usage in Template Definitions

13.2.1. Goals

`\IfExerciseGoalTF{`⟨*goal*⟩`}{`⟨*relation and value*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Checks the sum of goal ⟨*goal*⟩ against ⟨*relation and value*⟩.

`\IfExerciseGoalSingularTF{`⟨*goal*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Checks if the value of the goal ⟨*goal*⟩ of the current exercise equals 1. This is the same as `\IfExerciseGoalTF{`⟨*goal*⟩`}{=1}{`⟨*true*⟩`}{`⟨*false*⟩`}`.

3. The last sentence is wrong: those commands can be used anywhere but most of them only give useful results inside of templates.

\TotalExerciseTypeGoal{⟨goal⟩}{⟨type⟩}{⟨singular⟩}{⟨plural⟩}
Print the sum of goal ⟨goal⟩ for the exercises of type ⟨type⟩ and append ⟨singular⟩ or ⟨plural⟩ depending on wether the sum equals 1 or not.

\TotalExerciseGoal{⟨goal⟩}{⟨singular⟩}{⟨plural⟩}
Print the sum of goal ⟨goal⟩ for all exercises of all types and append ⟨singular⟩ or ⟨plural⟩ depending on wether the sum equals 1 or not.

13.2.2. Properties

* \IfExercisePropertyExistTF{⟨property⟩}{⟨true⟩}{⟨false⟩}
Tests wether an exercise property with the name ⟨property⟩ is defined.

\IfExercisePropertySetTF{⟨property⟩}{⟨true⟩}{⟨false⟩}
Tests wether the exercise property ⟨property⟩ has been set for the current exercise.

* \GetExerciseProperty{⟨property⟩}
Retrieves the value of the property ⟨property⟩ for the current exercise.

* \GetExerciseAliasProperty{⟨property⟩}
Retrieves the value of the property of which ⟨property⟩ is an alias of for the current exercise.

\SaveExerciseProperty{⟨property⟩}⟨macro⟩
Saves the value of the property ⟨property⟩ for the current exercise in macro ⟨macro⟩.

\GlobalSaveExerciseProperty
Globally saves the value of the property ⟨property⟩ for the current exercise in macro ⟨macro⟩.

\ExercisePropertyIfSetTF{⟨type⟩}{⟨id⟩}{⟨property⟩}{⟨true⟩}{⟨false⟩}
Test if the property ⟨property⟩ has been set for the exercise of type ⟨type⟩ with id ⟨id⟩.

* \ExercisePropertyGet{⟨type⟩}{⟨id⟩}{⟨property⟩}
Retrieves the value of the property ⟨property⟩ for the exercise of type ⟨type⟩ with id ⟨id⟩.

* \ExercisePropertyGetAlias{⟨type⟩}{⟨id⟩}{⟨property⟩}
Retrieves the value of the property of which ⟨property⟩ is an alias of for the exercise of type ⟨type⟩ with id ⟨id⟩.

\ExercisePropertySave{⟨type⟩}{⟨id⟩}{⟨property⟩}⟨macro⟩
Saves the value of the property ⟨property⟩ for the exercise of type ⟨type⟩ with id ⟨id⟩ in macro ⟨macro⟩.

\ExercisePropertyGlobalSave{⟨type⟩}{⟨id⟩}{⟨property⟩}⟨macro⟩
Globally saves the value of the property ⟨property⟩ for the exercise of type ⟨type⟩ with id ⟨id⟩ in macro ⟨macro⟩.

13.2.3. Parameters

* `\GetExerciseParameter{⟨parameter⟩}`
  Retrieves the value of the parameter ⟨*paramater*⟩ for the current exercise.

* `\GetExerciseName`
  Retrieves the value of the parameter `exercise-name` for the current exercise or of the parameter `solution-name` for the current solution.

* `\ExerciseParameterGet{⟨type⟩}{⟨id⟩}{⟨parameter⟩}`
  Retrieves the value of the parameter ⟨*parameter*⟩ for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩.

13.2.4. Tags

`\ForEachExerciseTag{⟨type⟩}{⟨code⟩}`
Loops over all tags of tag type ⟨*type*⟩ for the current exercise applying ⟨*code*⟩ each time. Inside ⟨*code*⟩ you can refer to the corresponding tag with `#1`.

`\ListExerciseTags{⟨type⟩}{⟨between⟩}`
Lists all tags of tag type ⟨*type*⟩ for the current exercise using ⟨*between*⟩ as a separator.

`\UseExerciseTags{⟨type⟩}{⟨between two⟩}{⟨between⟩}{⟨between last two⟩}`
Lists all tags of tag type ⟨*type*⟩ for the current exercise using ⟨*between*⟩ as a separator and ⟨*between last two*⟩ as separator between the last two tags of the list. If the list only consists of two tags ⟨*between two*⟩ is used as separator.

13.2.5. Further Commands for Usage in Template Definitions

* `\IfInsideSolutionTF{⟨true⟩}{⟨false⟩}`
  Tests if the template is used inside a solution environment or not.

`\ForEachPrintedExerciseByType{⟨code⟩}`
Loops over each *printed* exercise ordered by the exercise types and within each type by id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

#1: the type of the exercise

#2: the id of the exercise

#3: the counter of the exercise

#4: the subtitle of the exercise

#5: the points of the exercise

#6: the bonus points of the exercise

`\ForEachUsedExerciseByType{⟨code⟩}`
Loops over each *used* exercise ordered by the exercise types and within each type by id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

#1: the type of the exercise

#2: the id of the exercise

#3: the counter of the exercise

#4: the subtitle of the exercise

#5: the points of the exercise

#6: the bonus points of the exercise

`\ForEachPrintedExerciseByID`
Loops over each *printed* exercise order by the exercise id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

#1: the type of the exercise

#2: the id of the exercise

#3: the counter of the exercise

#4: the subtitle of the exercise

#5: the points of the exercise

#6: the bonus points of the exercise

`\ForEachUsedExerciseByID`
Loops over each *used* exercise order by the exercise id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

#1: the type of the exercise

#2: the id of the exercise

#3: the counter of the exercise

#4: the subtitle of the exercise

#5: the points of the exercise

#6: the bonus points of the exercise

* `\XSIMtranslate{`⟨*keyword*⟩`}`
Delivers the translation of ⟨*keyword*⟩ according to the current document language (in the meaning of a babel [Bra16] or polyglossia [Cha15] language). Existing keywords and keyword translations (and how to add new ones) are explained in section 14.

`\XSIMexpandcode{`⟨*code*⟩`}`
Expands ⟨*code*⟩ like `\edef` does and leaves the result in the input stream.

`\XSIMmixedcase{`⟨*code*⟩`}`
Converts ⟨*code*⟩ to mixed case:
`\XSIMmixedcase{this is some text}` This is some text

`\XSIMputright`⟨*macro*⟩`{`⟨*code*⟩`}`
Extends the macro definition of ⟨*macro*⟩ with ⟨*code*⟩ putting it to the right. This is like a local version of the LaTeX kernel macro `\g@addto@macro`.

* \XSIMifeqTF{⟨code 1⟩}{⟨code 2⟩}{⟨true⟩}{⟨false⟩}
  Checks if the full expansion[4] of ⟨code 1⟩ and ⟨code 2⟩ is the same tokenlist.

* \XSIMifblankTF{⟨code⟩}{⟨true⟩}{⟨false⟩}
  Checks if the full expansion[4] of ⟨code⟩ is blank (*i. e.*, if it is empty or only consists of spaces).

13.3. Declaring Templates

13.3.1. Environment Templates

\DeclareExerciseEnvironmentTemplate{⟨name⟩}{⟨begin code⟩}{⟨end code⟩}
Declare the environment template ⟨name⟩.

13.3.2. Heading Templates

\DeclareExerciseHeadingTemplate{⟨name⟩}{⟨code⟩}
Declare the heading template ⟨name⟩.

13.3.3. Grading Table Templates

\DeclareExerciseTableTemplate{⟨name⟩}{⟨code⟩}
Declare the grading table template ⟨name⟩.

13.4. Examples

14. Exercise Translations

15. Other Commands

A. All Exercise Examples

Exercise 1

A first example for an exercise.

Exercise 2 *This is a subtitle*

An exercise where some properties have been set.

Exercise 3

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit

4. This is a \romannumeral expansion [Flo].

purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

`\GetExerciseProperty{id}`: 3
`\GetExerciseAliasProperty{ID}`: 3
`\GetExerciseProperty{ID}`: 3

Exercise 4

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

`\GetExerciseProperty{id}`: 4
`\GetExerciseAliasProperty{ID}`: 4
`\GetExerciseProperty{ID}`: foo-bar

Exercise 5

This exercise will not be printed but the exercise counter will be incremented nonetheless. Its solution will be printed in the list of solutions.

Exercise 6

This exercise is added to the collection ‘foo’.

B. All Solution Examples

Solutions to the Exercises

Solution 1

A first example for a solution.

Solution 5

The solution of the exercise that has not been printed.

References

[Bra16] Johannes Braams, current maintainer: Javier Bezos.
babel. version 3.9q, Feb. 24, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/babel/.

[Cha15] François Charette, current maintainer: Arthur Reutenauer.
polyglossia. version 1.42.0, Aug. 6, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/polyglossia/.

[Fea05] Simon Fear. booktabs. version 1.61803, Apr. 14, 2005 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/booktabs/.

[Flo] Bruno Le Floch. *Cunning (La)TeX tricks.*
URL: http://tex.stackexchange.com/a/19769/ (visited on 03/02/2017).

[L3Pa] The LaTeX3 Project Team. l3kernel. version SVN 6377, Jan. 19, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/l3kernel/.

[L3Pb] The LaTeX3 Project Team.
l3packages. version SVN 6377, Jan. 19, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/l3packages/.

[Leh15] Philipp Lehman, current maintainer: Joseph Wright.
etoolbox. version 2.2a, Aug. 2, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox/.

[Nie15] Clemens Niederberger. translations. version 1.2e, Nov. 7, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Nie17] Clemens Niederberger. exsheets. version 0.21i, Feb. 8, 2017 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/exsheets/.

C. Index