WooWoo Specs

Tomáš Kalvoda, 2021-05-01, version 1.0alfa

Introduction

This document briefly describes the abstract structure of a WooWoo document and then one of its possible realizations employed at FIT CTU (the "FIT template"). The description presented here is organized in a top-down fashion.

The general idea behind the format is more or less finalized. In the future, there might be some minor tweaks, improvements, or changes in terminology, however. Changes in the FIT template and creation of new templates are expectable.

The main aim of the whole endeavor is to completely decouple the actual source code with all possible meta-information attached in a human-readable form and the presentation of the material.

Division of a WooWoo Document

Every WooWoo document can be divided into multiple parts, subparts, etc. Each *document part* has a *type*, *title*, and can be accompanied by a *meta-block* containing additional information.

The following piece of code illustrates the general structure of a document part.

```
.MyPart Title
  ...
  YAML meta-block
  ...
```

The first line of a document part begins with a dot, immediately continues with the part type beginning with a capital letter, and is terminated by the title of the document part. Optionally, the document part can be annotated by indented YAML block, a.k.a meta-block. There is no newline between the first line and the meta-block.

Available types of document parts are provided by the particular template used (see below). The handling of information given in a meta-block is defined by the template, too.

Blocks and Objects

The actual content of a WooWoo document is further divided into series of *blocks* and *objects*.

The following piece of code describes the general structure of a WooWoo *object*.

```
.ObjectType:
  ...
  meta-block
  ...


  indented block


  indented block


  ...
```

The first line of an object declaration starts with its *type* beginning with a capital letter and surrounded by a dot and a colon. The header can be immediately followed by an indented *meta-block* with additional information in YAML format. The main body of an object consists of indented *blocks* (see below) divided by *at least two empty lines.*

What exactly is a *block*? Block is a series of consistently indented *lines of text* (a text-block; possibly containing *inner environments* and annotated by indented meta-blocks), and *outer environments* separated by *single empty lines* (see below). An example of a typical simple block is presented below.

```
The "distributive law".notion.1 requires the equality
  1:
    index: law!distributive

.equation:
  label: eq-distributive-law

  a \times (b + c) = ab + ac

to hold for any $a,b,c \in \mathbb{R}$.
```

This block consists of two lines of text (the first and last line; the first line contains an annotated inner environment `.notion`) and outer environment (`.equation`). Inner and outer environments are described in more detail in the next section of this document.

Inner Environments

Inner environments enable the author to annotate parts of the text. There are two ways how to describe an inner environment.

Short form of an inner environment declaration has a prefix structure. It is demonstrated by the following piece of code:

```
... text text .innerEnvironment:body text text ...
```

The declaration begins with a dot followed by the environment type (its first letter is lower case) and colon, and the environment body *without any whitespace*.

Verbose form of the inner environment uses postfix structure and an optional reference to additional information attached in the meta-block. In this case the body is surrounded by quotation marks. So, the general structure is the following:

```
... text text ...
... text "body body".innerEnvironment.1 text ...
... text text ...
  1:
    key: value
```

The references to meta-blocks are numbered by arabic digits and appended to the environment type using a dot.

Inner environments are usually used when annotating small parts of the text (like emphasize, references, footnotes, citations, etc.).

Furthermore, there are two shorthanded versions of inner environments using `#` and `@` characters instead of dots. Their meaning depends on the template used, see bellow.

Outer Environments

Outer environments possibly carry larger content, like various types of equations, code blocks, etc. Their structure is similar to that of an object, the only difference is that the environment type begins with a lower case letter and the body is formed by only one block of text-lines. The following piece of code illustrates this concept.

```
.outerEnvironment:
  ...
  meta-block
  ...

  body
  body
  body
```

It is possible to skip the header part of the outer environment declaration. This is then interpreted as a shorthand for a particular outer environment defined by the template. In mathematically heavy texts this is used for entering equations (see `.equation` below). Alternatively, one can think of code blocks.

Outer environments can begin with a dot or a exclamation mark. Those beginning with an exclamation mark are thought to be *fragile* and WooWoo does not perform any transformation of its content. It is passed through template as-is.

Concrete examples of both inner and outer environments are provided below.

FIT Template as an example

Document Parts

FIT template structures the document into *Chapters*, *Sections*, and *Subsection*. At the moment these document parts are fairly simple. We use only one required meta-data field, `label`. This label is used for referencing of various parts and HTML templates stores Chapters and Sections in single files named according to those labels.

Few simple examples are provided below. These should be quite self-explanatory.

`.Chapter` Part

```
.Chapter Introduction
  label: chap-introduction
```

`.Section` Part

```
.Section Newton's method
  label: sec-newton
```

`.Subsection` Part

```
.Subsection Introduction
  label: chap-introduction
```

Objects

The FIT template considers blocks to be paragraphs of text.

The FIT template provides the following, mainly mathematical, objects.

`.Definition`

This is a simple mathematical definition of some notion. None of the keys in meta-block are obligatory. The author can specify `label` for future reference, `title` – usually the notion being defined –, and `index`. Index is used for building an index of notions and follows the usual LaTeX syntax of the `makeidx` LaTeX package.

```
.Definition:
  label: definition-label
  title: GCD
  index: definition!GCD

  Greatest common divisor (GCD) of nonzero positive integers $k$
  and $\ell$ is the largest integer that divides both $k$ and $\ell$.
```

`.Corollary`

Exactly the same comments as in the case of `.Definition` apply.

`.Lemma`

Exactly the same comments as in the case of `.Definition` apply.

`.Theorem`

Exactly the same comments as in the case of `.Definition` apply.

`.Proof`

This object holds a proof of a certain proposition. Meta-block can contain an optional `title` key.

```
.Proof:
  title: Proof of Riemann's Lemma

  Obvious.
```

`.Remark`

Exactly the same comments as in the case of `.Definition` apply.

`.Example`

Exactly the same comments as in the case of `.Definition` apply.

`.Figure`

Figure represents a labeled float holding pictures or diagrams. Its content is formed by a block with outer environments `!tikz` or `.image` and `.caption`. See below.

```
.Figure:
  label: myfigure

  !tikz:

    \draw[thick, ->] (0,0) -- (1,0) node[below] {$x$};

  .caption:

    A beautiful diagram!
```

`.Question`

This object enables the author sprinkle the text with question testing the readers understanding. The body of a question is formed by the actual question and an outer environment `.solution` holding the answer. The answer is hidden to the reader (HTML template requires the user to click, PDF template lists all answers at the back of the document).

```
.Question:

  Compute the following sum $1+1$.

  .solution:

    $2$.
```

`.Table`

Table represent a labeled float holding tabular data. Its content is formed by a block with outer environments `!tabular` and `.caption`. See below.

```
.Table:
  label: table1
  !tabular:

    a    b
    ---- ----
    1    2

  .caption:

    Table with two rows and columns.
```

Outer Environments

`.align`

`.align` is the WooWoo counterpart of AMSTex environment `align` used for sophisticated alignment of equations. Its content is just plain LaTeX code with optional math-popup annotation.

```
.align:

  f(x) &= x^2\,, \\
  g(x) &= \sin(x)\,.
```

`.caption`

Defines a caption of `.Table` or `.Figure`, see above.

`!codeblock`

Larger pieces of code listings can be described using the `.codeblock` outer environment. Optional meta-block can contain a `language` key which stores the language recognizable by the Pygments library.

```
!codeblock:
  code: python

  for k in range(1):
    print(k)
```

`.enumerate`

The `.enumerate` outer environment supports lists enumerated by digits or alphabet characters. By default we use arabic digits (1., 2., 3., etc.). Optionally, this behavior can be altered using the meta-block. Passing `a` under the `type` key will result in items being labeled by the roman letters a., b., c., etc.

For example,

```
.enumerate:
  type: a

  * First item.
  * Second item.
  * Third item.
```

will result in a list enumerated by roman letters.

See also the `.item` inner environment.

equation

The `.equation` inner environment represents a stand-alone mathematical equation, optionally labeled (with `label` key in the meta-block) for future reference and numbered. Supported code is the usual LaTeX/AMSTex markup.

```
.equation:
  label: eq-pythagoras

  a^2 + b^2 = c^2\,.
```

This code results in a single equation with label `eq-pythagoras`.

The FIT template defines anonymous outer environments (i.e. indented lines of text without header specs) to be `.equation.` So the following code

```
.equation:

  f(x) = \sin(x)\,.
```

is equivalent to

```
  f(x) = \sin(x)\,.
```

`!sage`

This outer environment can contain SageMath code which will be executed and the output will be passed in the WooWoo output.

`.solution`

Holds the answer to a question (see `.Question`).

`!tabular`

Contains table defined using the “simple tables” pandoc/markdown format.

`!tikz`

This fragile outer environment contains TikZ code of a figure.

Inner Environments

`.cite`

In order to refer the reader to an external source listed in the bibliography the author can use the `.cite` inner environment. The code `.cite:Einstein1915` renders the reference to article identified by key `Einstein1915` in the bibliography file. Bibliography entries are defined in an external text file and follow the usual BibTeX format.

`.code`

Inline code is rendered using the `.code` inner environment. For example, `"[ k for k in 1:10 ]".code` will result in something like `[ k for k in 1:10 ]`.

`.emphasize`

Text placed in the `.emphasize` inner environment is lightly emphasized in the output. FIT template uses italics for the emphasis.

`.eqref`

This a simple analog of the LaTeX `\eqref` macro. I.e. it typesets a reference to the numbered equation with a given label, for example `.eqref:newton-formula`. FIT thesis template numbers equations incrementally across chapters and surrounds the reference with the usual parenthesis. That is, (2.3) is the reference to the third numbered equation in the second chapter of the document. This numbering is invariant with respect to the output.

`.footnote`

This inner environment produces a footnote.

`.image`

Points to an external file that is to be inserted in the Figure float.

`.item`

Represents an item in a list (see `.enumerate` and `.itemize`). The author can use a simple short cut, the code

```
  * item content
```

is equivalent to

```
  "item content".item
```

`.itemize`

This is analogous to the `.enumerate`. The only difference that the items are not ordered and are delimited only by bullets.

`.math`

`.math` contains a piece of mathematical formula in LaTeX form. The author can of course use the usual LaTeX math delimiters, $. The code

```
$\sin(\pi)$
```

is equivalent to

```
"\sin(\pi).math"
```

`.notion`

This inner environment enables the author to highlight the notion being defined and store the definition in the index using the meta-block.

```
... we call such a function "increasing function".notion.1 ...
  1:
    index: function!increasing
```

`.quote`

Sometimes it is desirable to include a clever quotation in the text. This is supported by the `.quote` outer environment. Its meta-block has to specify the `author` of the quotation and optionally a `link` to the quotation source.

```
.quote:
  author: Some Clover Guy
  link: url://address

  Some very clever sentence.
```

`.quoted`

Represents quoted text with correct quotation marks (note that the Czech and English customs are different). I.e. `"blah".quoted` results in something like “blah”.

`.reference`

This inner environment enables the author to create references to labeled parts of the document or objects as in `.reference:newton-formula`

`.todo`

This is a simple ToDo note that the author can use when she is creating or editing the WooWoo document.

Shorthands

There are two shorthand version of the reference inner environment.

The first one uses the `#` delimiter followed by a label of an object: `"Newton's formula"#newton-formula`. This incantation will create an active link (clickable) named “Newton’s formula” to the equation or object with label `newton-formula`. In HTML output it will create a pop-up with the actual Newton’s formula.

The second one can be used when referencing external web page. Instead of

```
"beautiful page".reference.1"
  1:
    url: 'link'
```

one might use shorter

```
"beautiful page"@1"
  1: 'link'
```