Common Lisp Documentation Weaver

Mariano Montone (marianomontone@gmail.com)

Table of Contents

1 Introduction

CL-DOCWEAVER is a document weaver for Common Lisp.

Documentation for a Lisp project is written with the user's tool of choice (like Texinfo, Markdown, etc). Then, Common Lisp definitions are expanded into the documentation source using DocWeaver commands.

DOCWEAVER commands give the user control on how definitions are to be expanded, either via command options or by choosing a different set of commands.

CL-DOCWEAVER is easy to extend to support different documentation tools.

Texinfo and Markdown are the ones with best support at this moment.

2 Installation

3 Usage

First choose your documentation system (either Texinfo or Markdown at this moment).

Write your documentation as you normally would for the chosen documentation system; then, whenever you need to reference to something in your Common Lisp system, use DocWeaver commands to expand Lisp definitions.

Commands have the following syntax: `(@command-name &rest args)`.

For example, use `(@clfunction alexandria:flatten)` to expand the definition of `ALEXANDRIA:FLATTEN` function.

That looks like this:

`FLATTEN` (*tree*) [ALEXANDRIA]

Traverses the *tree* in order, collecting non-null leaves into a list.

Note that commands usually receive options in order to be able to control different aspects of the expanded definition.

By default, docstrings are parsed to extract possible references to other parts of the code; then those references are formatted as links that can be used to navigate the source code.

Finally, use See [DOCWEAVER:WEAVE-FILE function], page 5to weave your documentation system source files.

4 Commands

`@clvariable` (*function-symbol* **&rest** *args*) [Command]

Expands definition for variable bound to FUNCTION-SYMBOL.

For example,

```
(@clvariable cl:*standard-output*)
```

Looks like this:

`*STANDARD-OUTPUT*` [COMMON-LISP]

default output stream

`@clfunction` (*function-symbol* **&rest** *args*) [Command]

Expands definition for function bound to FUNCTION-SYMBOL.

For example,

```
(@clfunction alexandria:map-permutations)
```

Looks like this:

`MAP-PERMUTATIONS` (*function sequence* **&key** *(start 0) end length (copy t)*) [ALEXANDRIA]

Calls *function* with each permutation of *LENGTH* constructable from the subsequence of *SEQUENCE* delimited by *START* and *END*. *START* defaults to 0, *END* to *length* of the *sequence*, and *LENGTH* to the *length* of the delimited subsequence.

`@clpackage` (*package-name* **&key** *(include-external-definitions t) include-internal-definitions (categorized t)*) [Command]

Displays PACKAGE-NAME package description.

If INCLUDE-EXTERNAL-DEFINITIONS is T, then all package external definitions are shown.

If INCLUDE-INTERNAL-DEFINITIONS is T, then all package internals definitions are shown.

If CATEGORIZED is T, then definitions appear categorized in sections (variables, functions, etc).

Example:

```
(@clpackage :alexandria)
```

`@clref` (*symbol type*) [Command]

Creates a reference to SYMBOL. TYPE should be one of 'variable', 'function', 'class', etc.

Example:

```
(@clref alexandria:flatten, function)
```

5 API

DOCWEAVER [PACKAGE]

External definitions

Functions

WEAVE-FILE (*file output-file* **&rest** *options* **&key** *docsystem modules command-prefix (parse-docstrings t)*) [DOCWEAVER]

Weaves documentation source in *FILE* and writes the result to *OUTPUT-FILE*.

Arguments:

- DOCSYSTEM: specify the documentation tool that is being used (*:texinfo*, *:markdown*, etc.).
- MODULES: is the list of *modules* (or ASDF system names) that need to be loaded to be able to read definition descriptions.
- COMMAND-PREFIX: is the character to use as prefix for commands. The character ‘at‘ is the default.
- PARSE-DOCSTRINGS: if *T*, then docstings are parsed and highlighted and references to code from it created.

6 Index

(Index is nonexistent)