shsite2 manual

eelvex

June 25, 2012

Contents

Chapter 1

How it works

1.1 variables, parse-commands and functions

Define variable or function: "- key:"

Variables: `%{name}%`

Functions: `%{name: args}%`

Parse-Commands: `%{!command}%`

Includes: `%{@file}%`

Any nest level:

`%{name-%{!com: args}%-%{@%{!echo "file"}%}% }%` will first run command "com" with arguments "args", then will run "echo "file"", will include "file" which will make the name of the variable: "name-xxx-xxx"

Parse-commands and includes are expanded immediately (leaving a warning string if not found) while variables and functions keep their form until they can be evaluated - if ever.

Differences between parse-commands and functions

parse-commands are more robust: no need for character escaping etc

parse-commads are always known so always evaluated. functions are context-depended.

functions can be treated like variables, so they are better suited for passing to other functions (like `map`) or parse-commands.

1.2 Makefile

Makefile variables:

`SITE_sources:` files to be parsed

`SITE_files:` files to be used as are (copied)

`SITE_target_dir:` output directory for this site. Default value is `SITE/`

`SITE_indexes:` files used for indexing and site-map. Default value is to point to `SITE_sources`.

Dynamic template with default, example

```
define eelvex.net_rule =
    @echo "Building ${2}"
    @mkdir -p $$(dirname ${2})
    @template=$$(shsite2 get-key ${1} template); \
        shsite2 parse preconfig.slc ${1} $${template:-template.html} | shsite2 parse p
endef
```

Chapter 2

Usage/examples

Chapter 3

Commands

3.1 *make*

make [site]
    Invoke gnu-make command after some basic checks.
    A default Makefile is provided that is supposed to build a site from your sources.

3.2 *publish*

publish [site]
    Publish your site through rsync,git,(s)ftp or other user-specified method.

3.3 *info*

info ¡path — name — fullpathname — title — depends¿ ¡file¿
    Get relevant info from file.

3.4 get-key

3.5 blocks

3.6 block-put

3.7 parse

Chapter 4

parse-commands

Use the first line on commands that take multi-line input, for options. Separate options with spaces.

and configuration variables point to paths where parse-commands reside (default and user respectively).

4.1 table-tex

4.2 code

Chapter 5

library functions

5.1 standard %{@stdfunctions}%

map

```
%{map:  '%{func}%' arg1 arg2 'arg3-a arg3-b' ...}%
```

is the same as:

```
%{func:  arg1}%
%{func:  arg2}%
%{func:  arg3-a arg3-b}%
...
```

rapply

if, if-set

5.2 web %{@webfunctions}%

link

in-url, in-title

5.3 db %{@dbfunctions}%

db-list-posts