SKILL#

Cadence SKILL++ Enhanced Framework

Targeted Audience : Anyone who writes SKILL

Created by [Aurélien Buchet]

Open Source Repository on [GitHub]

Introduction

Out-of-the-Box Tools

- Advanced Linter
- Code formatter
- Enhanced Finder

Advanced SKILL++ Features

- Unit-Testing Framework
- Documentation Generation
- On-Demand Type-Checking
- Fully Tested API

Requirements

To Use

Virtuoso with `$CDS_INST_DIR` defined

To Develop

- GNU Make
- shellcheck ≥ 0.10.0
- shellspec ≥ 0.28.1
- scc
- fd

Standalone Usage

`$SKILL_SHARP_ROOT/bin/sharp` commands:

Command	Documentation
`sharp help`	Display available commands and arguments.
`sharp lint`	Run advanced Linter on provided files.
`sharp test`	Load files and print test report.
`sharp globals`	Load files and report global definitions.
`sharp docgen`	Load files and print associated `.fnd` content.

Usage Inside Virtuoso

In the CIW or in `.cdsinit` to enable all SKILL# features:

```
(load (simplifyFilename "$SKILL_SHARP_ROOT/skill/loader.scm"))
```

`$SKILL_SHARP_ROOT` should point to the installation root

Features

Advanced Linter

Waiver

Detect unused local functions

Detect superseded functions

Hints

Formatter

Type-Checking

On-Demand

```
(@fun join
  ( ( char    ?type symbol|string         ) ; char is a symbol or a string
    @rest
    ( strings ?type ( symbol|string ... ) ) ; strings is a list of symbols or strings
    )
  ?doc "Join STRINGS using CHAR as junction character."
  ?out string                               ; output is a string
  (buildString strings char)
  )
```

To enable strict type-checking:

Set `$SKILL_SHARP_STRICT_TYPE_CHECKING` to TRUE

Type-Checking

Always Disabled

```
;; Type-checking always disabled
;; (Still useful for documentation)

(@fun convert_to_string
  ( ( name ?type symbol )
    )
  ?doc    "Convert NAME to a string and return it."
  ?out    string
  ?strict nil                                  ; Type-checking is disabled here
  (strcat name)
  )
```

Type-Checking

Always Enabled

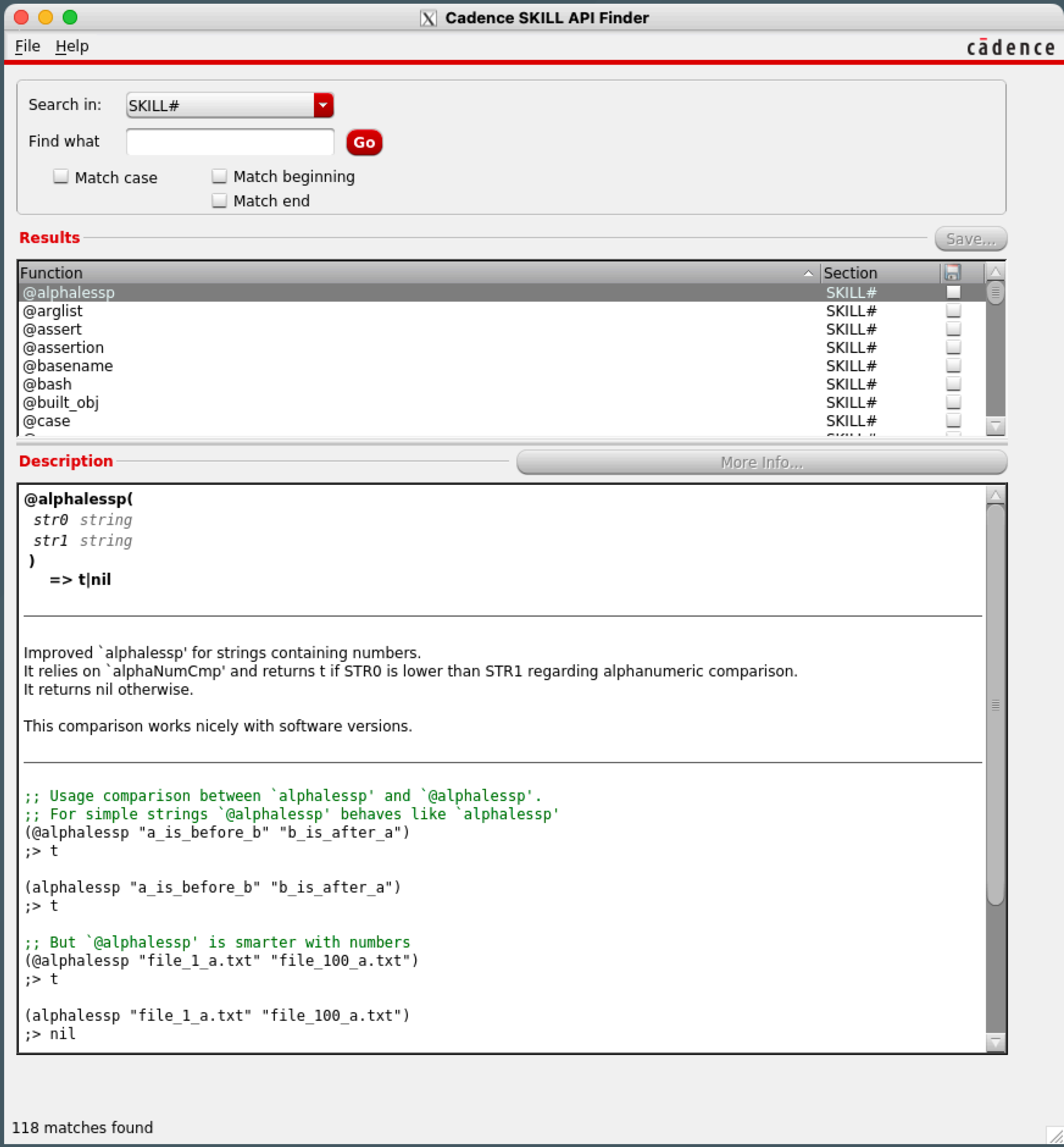
```
;; Type-checking always enabled
;; (Useful for top functions)

(@fun double
  ( ( num ?type float|integer )
    )
  ?doc    "Return the double of NUM as a float."
  ?out    float
  ?strict t                                   ; Type-checking is forced here
  ( num * 2.0 )
  )
```

Documentation Generation

Enhanced Finder

- Search in name, description, ...
- Match PCREs
- Case-sensitive when upper case
- Restrict to name, description, ...



Function	Section
@alphalessp	SKILL#
@arglist	SKILL#
@assert	SKILL#
@assertion	SKILL#
@basename	SKILL#
@bash	SKILL#
@built_obj	SKILL#
@case	SKILL#

```
@alphalessp(
 str0 string
 str1 string
)
  => t|nil
```

Improved `alphalessp' for strings containing numbers.
It relies on `alphaNumCmp' and returns t if STR0 is lower than STR1 regarding alphanumeric comparison.
It returns nil otherwise.

This comparison works nicely with software versions.

```
;; Usage comparison between `alphalessp' and `@alphalessp'.
;; For simple strings `@alphalessp' behaves like `alphalessp'
(@alphalessp "a_is_before_b" "b_is_after_a")
;> t

(alphalessp "a_is_before_b" "b_is_after_a")
;> t

;; But `@alphalessp' is smarter with numbers
(@alphalessp "file_1_a.txt" "file_100_a.txt")
;> t

(alphalessp "file_1_a.txt" "file_100_a.txt")
;> nil
```

118 matches found

F-Strings

```
;; Values are placed where they should be evaluated:
(@str "The result of 12 plus 27 is {12+27}.)")

(@str "
The current shell is {(getShellEnvVar \"SHELL\")}.
The current time is {(getCurrentTime)}.
")

;; `printf` formatting is supported
pi = (acos -1)
(@str "
This is pi with four decimals: {pi%0.4f}. \n\
The default is {pi}. \n\
")
```

Letf

```
;; `simplifyFilename` breaks if `(rexMagic)` is nil.
(progn (rexMagic nil) (simplifyFilename "$SHELL"))

;; `@letf` uses `setf` to set anything temporarily.
(@letf ( ( (rexMagic) t ) ) (simplifyFilename "$SHELL"))
(rexMagic) ; `(rexMagic)` value is still nil.

;; `@letf` works with any `setf` helper
(@letf ( ( (rexMagic)                             nil    )
         ( (getShellEnvVar "CUSTOM_VARIABLE") "TRUE" )
         ( (status optimizeTailCall)            t      )
         )
  (list (rexMagic) (getShellEnvVar "CUSTOM_VARIABLE") (status optimizeTailCall))
  )
```