Package ‘slash’

April 18, 2025

Type Package

Title Path-Based Access and Manipulation of Nested Lists

Version 0.1.0

Description Allows users to list data structures using path-based navigation.
Provides intuitive methods for storing, accessing,
and manipulating nested data through simple path strings.
Key features include strict mode validation, path existence checking, recursive
operations, and automatic parent-level creation. Designed for use cases requiring
organized storage of complex nested data while maintaining simple access patterns.
Particularly useful for configuration management, nested settings, and any
application where data naturally forms a tree-like structure.

License MIT + file LICENSE

Encoding UTF-8

URL `https://github.com/feddelegrand7/slash`

BugReports `https://github.com/feddelegrand7/slash/issues`

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.1

Imports R6

NeedsCompilation no

Author Mohamed El Fodil Ihaddaden [aut, cre]

Maintainer Mohamed El Fodil Ihaddaden <`ihaddaden.fodeil@gmail.com`>

Repository CRAN

Date/Publication 2025-04-18 13:10:01 UTC

Contents

`slash` *Path-based access and manipulation for R lists*

Description

Path-based access and manipulation for R lists

Path-based access and manipulation for R lists

Details

The slash class provides tools for working with hierarchical R lists using path-like strings (e.g., "a/b/c"). Rather than creating a new data structure, it adds convenient path-based access methods to standard R lists, supporting: - Both named and numeric (1-based) indexing - Strict mode for error checking - Various operations for list manipulation

Methods

Public methods:

- `slash$new()`
- `slash$get()`
- `slash$set()`
- `slash$exists()`
- `slash$delete()`
- `slash$clear()`
- `slash$get_all()`
- `slash$print()`
- `slash$print_list()`
- `slash$list_paths()`
- `slash$is_strict()`
- `slash$set_strict()`
- `slash$clone()`

Method `new()`**:** Create a new slash object

Usage:

```
slash$new(data = list(), strict = FALSE)
```

Arguments:

`data` Initial data (must be a list)

`strict` If TRUE, attempts to access non-existent paths will error

Returns: A new ‘slash‘ object

Method `get()`**:** Get value at specified path

Usage:

```
slash$get(path = NULL, default = NULL)
```

Arguments:

`path` Path to the element (e.g., "a/b/c" or "1/2/3")

`default` Value to return if path doesn't exist (NULL by default)

Returns: The value at the specified path, or default if not found

Method `set()`**:** Set value at specified path

Usage:

```
slash$set(path, value)
```

Arguments:

`path` Path to the element

`value` Value to set

Returns: The slash object (invisibly) for chaining

Method `exists()`**:** Check if path exists

Usage:

```
slash$exists(path)
```

Arguments:

`path` Path to check

Returns: TRUE if path exists, FALSE otherwise

Method `delete()`**:** Delete element at specified path

Usage:

```
slash$delete(path)
```

Arguments:

`path` Path to delete

Returns: The slash object (invisibly) for chaining

Method `clear()`**:** Clear all data

Usage:

```
slash$clear()
```

Returns: The slash object (invisibly) for chaining

Method `get_all()`**:** Get all data as a list

Usage:

```
slash$get_all()
```

Returns: The complete data structure

Method `print()`**:** Print summary of slash object

Usage:

```
slash$print(show_full = FALSE)
```

Arguments:

`show_full` If TRUE, shows full structure (FALSE by default)

Method `print_list()`**:** Print list structure at path

Usage:

```
slash$print_list(path = NULL)
```

Arguments:

`path` Path to print (NULL for root)

Method `list_paths()`**:** List all available paths

Usage:

```
slash$list_paths()
```

Returns: Character vector of all paths in the data structure

Method `is_strict()`**:** Check if in strict mode

Usage:

```
slash$is_strict()
```

Returns: TRUE if in strict mode, FALSE otherwise

Method `set_strict()`**:** Set strict mode

Usage:

```
slash$set_strict(strict)
```

Arguments:

`strict` Logical value for strict mode

Returns: The slash object (invisibly) for chaining

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
slash$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index