Package ‘fig’

October 13, 2022

Title  A Config Package with No "Con"
Version  1.0.0
Description  Work with configs with a source precedence. Either create own R6 instance or work with convenient functions at a package level.

URL  https://github.com/TymekDev/fig
BugReports  https://github.com/TymekDev/fig/issues
License  MIT + file LICENSE
Encoding  UTF-8
RoxygenNote  7.1.2
Imports  R6
Suggests  covr, testthat (>= 3.0.0), withr
Config/testthat/edition  3
NeedsCompilation  no
Author  Tymoteusz Makowski [cre, aut]
Maintainer  Tymoteusz Makowski <makowski.tymoteusz@gmail.com>
Repository  CRAN
Date/Publication  2022-03-31 13:50:02 UTC

R topics documented:

Fig ................................................................. 2
fig_configure .................................................... 7
fig_delete ....................................................... 8
fig_get .......................................................... 9
fig_store ....................................................... 10

Index 11
Description

Fig class is a main driver of this package. For usage details refer to Fig class methods documentation.

Fig provides a set of exported functions. This makes Fig class instance creation optional, and makes the package itself mimic being a class instance. Those functions are wrappers on an internal Fig object.

Methods

Public methods:

- Fig$new()
- Fig$configure()
- Fig$delete()
- Fig$delete_all()
- Fig$get()
- Fig$get_many()
- Fig$get_all()
- Fig$store()
- Fig$store_list()
- Fig$store_many()
- Fig$clone()

Method new(): Create a New Fig Instance

Usage:

Fig$new(env_prefix = "", split_on = ".")

Arguments:

env_prefix (character) A prefix to be prepended to a key before system environment lookup.
split_on (character) A value to split keys on. See Details section. Providing an empty string disables this behavior.

Details:  Fig treats character provided in split_on as key nest level delimiter. Therefore, split_on set to "." (default value) fig$get("foo.bar") is equivalent to fig$get("foo")$bar. Similarly fig$set("foo.bar", 1) is equivalent to fig$set("foo", list(bar = 1)). This behavior can be disabled either by passing an empty string either to new() during Fig instance creation or to configure() function to modify an existing instance.

Returns: New instance of Fig.

Examples:

fig <- Fig$new()
fig <- Fig$new(env_prefix = "RCONNECT_")
Method configure(): Configure a Fig Instance

Usage:
Fig$configure(env_prefix, split_on)

Arguments:
env_prefix (character) A prefix to be prepended to a key before system environment lookup. Pass an empty string to reset.
split_on (character) A value to split keys on. See Details section in new(). Providing an empty string disables this behavior.

Details: Unset arguments do not change configuration.

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new(env_prefix = "RCONNECT_")
fig$configure(env_prefix = "foo_")
fig$configure(split_on = "")
fig$configure() # has no effect

Method delete(): Delete Stored Values

Usage:
Fig$delete(...)

Arguments:
... Keys to be deleted.

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new()
fig$store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig$delete("foo")
fig$delete("bar", "baz")
fig$get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)

Method delete_all(): Delete All Stored Values

Usage:
Fig$delete_all()

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new()
fig$store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig$delete_all()
fig$get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)

Method get(): Retrieve a Stored Value

Usage:
Fig$get(key)
Arguments:

key  A key to retrieve a value for.

Details: This function returns values based on a following priority (highest to lowest). If value is not found, then it looks up next level in the precedence.

1. System environment variable (case sensitive)
2. Value manually set

For system environment lookup dots are replaced by underscores, e.g. `fig$get("foo.bar")` will look up `foo_bar`.

Returns: A value associated with provided key.

Examples:

```r
fig <- Fig$new()
fig$store("foo", 1)
fig$get("foo")

fig$store("bar", list(baz = 2))
fig$get("bar.baz")

fig$configure(split_on = "")
fig$get("bar.baz") # == NULL
```

Method `get_many()`: Retrieve Any Number of Stored Values

Usage:

```r
Fig$get_many(...)```

Arguments:

... Keys to retrieve values for.

Details: See `get()` Details section.

Returns: An unnamed list of values associated with keys provided in ....

Examples:

```r
fig <- Fig$new()
fig$store_many(foo = 1, bar = 2, baz = 3)
fig$get_many("foo", "bar")
```

Method `get_all()`: Retrieve All Stored Values

Usage:

```r
Fig$get_all()
```

Details: See `get()` Details section.

Returns: An unnamed list of all stored values.

Examples:

```r
fig <- Fig$new()
fig$store_many(foo = 1, bar = 2, baz = 3)
fig$get_all()
```

Method `store()`: Store a Value
Usage:
Fig$store(key, value)

Arguments:
key  A key to store a value for.
value A value to be stored.

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new()
fig$store("foo", 1)
fig$store("bar", 123)$store("baz", list(1, 2, 3))
fig$store("x.y", "a")

Method store_list(): Store a List’s Contents

Usage:
Fig$store_list(l)

Arguments:
l (named list) Names are used as keys for storing their values.

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new()
fig$store_list(list(foo = 123, bar = "abc"))

Method store_many(): Set Any Number of Values

Usage:
Fig$store_many(...) 

Arguments:
... Named arguments. Names are used as keys for storing argument values.

Returns: Reference to self. Other methods can be chained after this one.

Examples:
fig <- Fig$new()
fig$store_many("foo" = 1, "bar" = 2)
fig$store_many("foo.bar.baz" = 1)
fig$store_many("foo" = "a", "baz" = 123)

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

Usage:
Fig$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.
Examples

```r
## Method `Fig$new`

fig <- Fig$new()
fig <- Fig$new(env_prefix = "RCONNECT_")

## Method `Fig$configure`

fig <- Fig$new(env_prefix = "RCONNECT_")
fig$configure(env_prefix = "foo_")
fig$configure(split_on = "")
fig$configure() # has no effect

## Method `Fig$delete`

fig <- Fig$new()
fig$store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig$delete("foo")
fig$delete("bar", "baz")
fig$get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)

## Method `Fig$delete_all`

fig <- Fig$new()
fig$store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig$delete_all()
fig$get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)

## Method `Fig$get`

fig <- Fig$new()
fig$store("foo", 1)
fig$get("foo")
fig$store("bar", list(baz = 2))
fig$get("bar.baz")
fig$configure(split_on = "")
fig$get("bar.baz") # == NULL
```
fig_configure

Configure the Global Fig Instance

Description

This function allows modifying configuration of the global fig. Refer to argument descriptions for available options.

Usage

fig_configure(env_prefix, split_on)
Arguments

env_prefix (character) A prefix to be prepended to a key before system environment lookup. Pass an empty string to reset.
split_on (character) A value to split keys on. See Details section in Fig. Providing an empty string disables this behavior.

Details

Unset arguments do not change configuration.

Value

Reference to the global fig instance. Other methods can be chained after this one.

Examples

```r
fig_configure(env_prefix = "foo-")
fig_configure(split_on = "")
fig_configure() # has no effect
```

---

fig_delete

Delete Stored Values

Description

These functions allow deleting values stored in the global fig instance.

Usage

```r
fig_delete(...) 
fig_delete_all()
```

Arguments

... Keys to be deleted.

Value

Reference to the global fig instance. Other methods can be chained after this one.

Examples

```r
fig_store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig_delete("foo")
fig_delete("bar", "baz")
fig_get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)
fig_store_many("foo" = 1, "bar" = 2, "baz" = 3)
fig_delete_all()
fig_get_many("foo", "bar", "baz") # == list(NULL, NULL, NULL)
```
fig_get

Retrieve Stored Values

Description

These functions allow retrieving values stored in the global fig instance.

Usage

fig_get(key)

fig_get_many(...)

fig_get_all()

Arguments

key

A key to retrieve a value for.

... 

Keys to retrieve values for.

Details

These functions return values based on a following priority (highest to lowest). If value is not found, then it looks up next level in the precedence.

1. System environment variable (case sensitive)
2. Value manually set

For system environment lookup dots are replaced by underscores, e.g. `fig_get("foo.bar")` will look up `foo_bar`.

Value

A value associated with provided key.

An unnamed list of values associated with keys provided in ...

An unnamed list of all stored values.

Examples

fig_store("foo", 1)
fig_get("foo")

fig_store("bar", list(baz = 2))
fig_get("bar.baz")

fig_configure(split_on = "")
fig_get("bar.baz") # == NULL
fig_store_many(foo = 1, bar = 2, baz = 3)
fig_get_many("foo", "bar")
fig_store_many(foo = 1, bar = 2, baz = 3)
fig_get_all()

---

**fig_store**

---

### Store Values

**Description**

These functions allow storing values in the global fig instance.

**Usage**

- `fig_store(key, value)`
- `fig_store_list(l)`
- `fig_store_many(...)`

**Arguments**

- **key**
  - A key to store a value for.
- **value**
  - A value to be stored.
- **l**
  - (named list) Names are used as keys for storing their values.
- **...**
  - Named arguments. Names are used as keys for storing argument values.

**Value**

Reference to self. Other methods can be chained after this one.

**Examples**

- `fig_store("foo", 1)`
- `fig_store("bar", 123)`
- `$store("baz", list(1, 2, 3))`
- `fig_store("x.y", "a")`
- `fig_store_list(list(foo = 123, bar = "abc"))`
- `fig_store_many("foo" = 1, "bar" = 2)`
- `fig_store_many("foo.bar.baz" = 1)`
- `fig_store_many("foo" = "a", "baz" = 123)"`
Index

Fig, 2, 8
fig_configure, 7
fig_delete, 8
fig_delete_all (fig_delete), 8
fig_get, 9
fig_get_all (fig_get), 9
fig_get_many (fig_get), 9
fig_store, 10
fig_store_list (fig_store), 10
fig_store_many (fig_store), 10