Package 'erratum'

January 3, 2022

Title Handle Error and Warning Messages

Version 2.2.0

Description Elegantly handle error and warning messages.

License AGPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Imports R6, rlang

Suggests testthat, covr

Config/testthat/edition 3

BugReports `https://github.com/devOpifex/erratum/issues/`

NeedsCompilation no

Author John Coene [aut, cre],
Opifex [cph]

Maintainer John Coene `<john@opifex.org>`

Repository CRAN

Date/Publication 2022-01-03 21:40:02 UTC

R topics documented:

`bash` *Take a Bash*

Description

Equivalent to `tryCatch()`.

Usage

```
bash(expr, e = NULL, w = NULL)
```

Arguments

`expr`	Expression to run, passed to `tryCatch()`.
`e, w`	An object of class `Error` or `Warning` as returned by `e()` or `w()`.

Examples

```
safe_log <- function(x){
 result <- bash(log(x))

 if(is.e(result))
   stop(result$stop())

 return(result)
}

if(interactive())
 safe_log("a")
```

`checks` *Check*

Description

Check whether an object is an error or a warning.

Usage

```
is.e(obj)

## Default S3 method:
is.e(obj)

## S3 method for class 'err'
is.e(obj)

is.w(obj)

## Default S3 method:
is.w(obj)

## S3 method for class 'err'
is.w(obj)

is.problem(obj)
```

Arguments

`obj`	Object to check.

Value

A boolean value.

Functions

- `is.e`: Whether the object is an error.
- `is.w`: Whether the object is a warning.
- `is.problem`: Whether the object is an error or a warning.

Examples

```
err <- e("Whoops!")

is.e(err)
is.w(err)
```

`chk` *Check*

Description

Checks individual objects.

Usage

```
chk(obj)

## Default S3 method:
chk(obj)

## S3 method for class 'err'
chk(obj)
```

Arguments

`obj`	Object to check.

Details

Runs `warning()` or `stop()` where necessary.

`Error` *Error*

Description

Error

Error

Super class

`erratum::Issue` -> `Error`

Methods

Public methods:

- `Error$new()`
- `Error$stop()`
- `Error$fatal()`
- `Error$clone()`

Method `new()`**:**

Usage:

```
Error$new(obj)
```

Arguments:

`obj` A character string or an object of class `error`, or `warning`.

`type` Type of message.

Details: Initialise

Method `stop()`**:**

Usage:

```
Error$stop()
```

Details: Stop
Analogous to `stop()`

Method `fatal()`**:**

Usage:

```
Error$fatal()
```

Details: Fatal
Analogous to `stop()`

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ew` *Handlers*

Description

Handle errors and warnings.

Usage

```
e(obj)

w(obj)
```

Arguments

`obj`	A character string or an object of class `error`, or `warning`.

Examples

```
err <- e("Something went wrong")

foo <- function(x){
 if(is.character(x))
   return(err)

 log(x)
}

foo("a")
```

`Issue` *Core Class*

Description

Core class to create and handle issues.

Active bindings

`rule` Rules to perform checks, must be functions that accept a single argument and return a boolean.

`message` The message (warning or error).

`call` Expression or function (as string) that led to the issue.

`raiser` Function to run when the `raise` method is called. By default the error uses `stop()` and warning uses `warning()`. The function must accept a single argument: the error message (character vector).

Methods

Public methods:

- `Issue$new()`
- `Issue$print()`
- `Issue$return()`
- `Issue$addRule()`
- `Issue$check()`
- `Issue$raise()`
- `Issue$clone()`

Method `new()`**:**

Usage:

```
Issue$new(obj, type = c("error", "warning"))
```

Arguments:

`obj` A character string or an object of class `error`, or `warning`.

`type` Type of message.

Details: Initialise

Method `print()`**:**

Usage:

```
Issue$print()
```

Details: Print
Print message of error or warning.

Method `return()`**:**

Usage:

```
Issue$return(n = 1)
```

Arguments:

`n` the number of generations to go back, passed to `parent.frame()`.

Details: Return Returns self from parent function.

Method `addRule()`**:**

Usage:

```
Issue$addRule(fn)
```

Arguments:

`fn` Function defining rule, must accept a single argument and return a boolean.

Details: Add a rule

Method `check()`**:**

Usage:

```
Issue$check(obj)
```

Arguments:

`obj` Object to check by rules

Details: Add a predicate

Method `raise()`**:**

Usage:

```
Issue$raise(fn = NULL)
```

Arguments:

`fn` A function to use to raise the issue.

Details: Raise error or warning

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

latch *Latch an Error or a Warning*

Description

Latch an error or a warning to an object to indicate an issue associated with it. These can later be checked with `is.e()` and `is.w()`, and can also be `resolve()`.

Usage

```
latch.e(obj, error)

latch.w(obj, warning)

unlatch(obj)
```

Arguments

`obj`	Object to latch the error or warning onto.
`error`, `warning`	Error or warning, the output of `e()` or `w()`.

Functions

- `latche` and `latchw`: latch an error or a warning.
- `unlatch`: unlatch any error or warning.

Examples

```
x <- 1
problematic <- latch.e(x, e("Not right"))

is.e(problematic)

do_sth_with_x <- function(x){
 resolve(x)
 x + 1
}

if(interactive()){
 do_sth_with_x(x)
 do_sth_with_x(problematic)
}

unlatch(problematic)
```

raise *Raisers*

Description

Set `raise` method globally, every subsequent `raise` method will make use of this function.

Usage

```
raise.e(fn = NULL)

raise.w(fn = NULL)
```

Arguments

`fn`	Function to run when the `raise` method is called. By default the error uses `stop()` and warning uses `warning()`. The function must accept a single argument: the error message (character vector).

resolves *Resolve Errors and Warnings*

Description

Resolve Errors and Warnings

Usage

```
resolve(...)

defer_resolve(...)
```

Arguments

`...`	Objects to check, if any of them is an `Error` then `stop()` is called, if any are `Warnings` then `warning()` is called.

Details

Objects passed are evalutated in order.

Value

Invisiby returns `NULL`

skip — *Skip*

Description

Skip the rest of the function; calls `return()` in the parent function if any object is an error or (optionally) a warning.

Usage

```
skip(..., w = FALSE)
```

Arguments

`...`	Objects to check, if any of them is an `Error` then it calls `return()` in the parent function, this can optionally be applied if any object is a `Warning` with the `w` argument.
`w`	Whether to also skip is there are `Warning`.

template — *Templates*

Description

Define error and warning templates.

Usage

```
template.e(pat = "%s")

template.w(pat = "%s")
```

Arguments

`pat`	Pattern to use, must include %s, forwarded to `sprintf()`.

Examples

```
msg <- "Something's wrong"

# default
e(msg)

# template
template.e("Whoops: %s - sorry!")
e(msg)
```

```
# reset
template.e()
```

`Warning` *Error*

Description

Error

Error

Super class

`erratum::Issue` -> `Warning`

Methods

Public methods:

- `Warning$new()`
- `Warning$warn()`
- `Warning$clone()`

Method `new()`**:**

Usage:

`Warning$new(obj)`

Arguments:

`obj` A character string or an object of class `error`, or `warning`.

`type` Type of message.

Details: Initialise

Method `warn()`**:**

Usage:

`Warning$warn()`

Details: Warn
Analogous to `warning()`

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

Usage:

`Warning$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Index