# Package 'log'

October 13, 2022

**Title** Record Events and Issues

**Version** 1.1.1

**Description** Logger to keep track of informational events and errors useful for debugging.

**License** AGPL-3

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Imports** R6, cli, crayon

**Suggests** shiny, plumber, testthat (>= 3.0.0), covr

**Config/testthat/edition** 3

**BugReports** https://github.com/devOpifex/log/issues/

**NeedsCompilation** no

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

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

**Repository** CRAN

**Date/Publication** 2022-02-24 19:40:02 UTC

## R topics documented:

| inputLogger | *Input logger* |
|---|---|

### Description

Default logger used to log inputs in [logApp()](logApp()).

### Usage

```
inputLogger
```

### Format

An object of class Logger (inherits from R6) of length 13.

| is.log | *Log Check* |
|---|---|

### Description

Log Check

### Usage

```
is.log(obj)
```

### Arguments

obj             Object to check.

### Value

TRUE if object is a logger, and FALSE otherwise.

### Examples

```
info <- Logger$new("INFO")
is.log(info)
```

---

Logger *Logger*

---

**Description**

Create a logger.

**Public fields**

printer A callback function to write the message to the console, must accept a single argument, defaults to cat.

predicate A predicate function that determines whether to actually run the log, useful if you want to switch the logger on and off for debugging.

If the function returns TRUE the logger runs as normal, if FALSE the logger does not actually print, write or dump the messages.

**Methods**

**Public methods:**

- [Logger$new()](#)
- [Logger$date()](#)
- [Logger$time()](#)
- [Logger$unix()](#)
- [Logger$hook()](#)
- [Logger$dir()](#)
- [Logger$flag()](#)
- [Logger$log()](#)
- [Logger$dump()](#)
- [Logger$clone()](#)

**Method** new():

*Usage:*

```
Logger$new(prefix = NULL, write = FALSE, file = "log.log", sep = "\t")
```

*Arguments:*

prefix String to prefix all log messages.

write Whether to write the log to the file.

file Name of the file to dump the logs to, only used if write is TRUE.

sep Separator between prefix and other flags and messages.

*Details:* Initialise

*Examples:*

```
info <- Logger$new("INFO")
info$log("hello")
```

**Method** `date()`:

*Usage:*

```
Logger$date(format = "%d-%m-%Y")
```

*Arguments:*

`format` Formatter for the item, passed to [format()](#).

*Details:* Include the date in the log

*Examples:*

```
info <- Logger$new("INFO")$date()
info$log("today")
```

**Method** `time()`:

*Usage:*

```
Logger$time(format = "%H:%M:%S")
```

*Arguments:*

`format` Formatter for the item, passed to [format()](#).

*Details:* Include the time in the log

*Examples:*

```
info <- Logger$new("INFO")$time()
info$log("now")
```

**Method** `unix()`:

*Usage:*

```
Logger$unix()
```

*Details:* Include the time in the log

*Examples:*

```
info <- Logger$new("INFO")$unix()
info$log("timestamp")
```

**Method** `hook()`:

*Usage:*

```
Logger$hook(fn)
```

*Arguments:*

`fn` A function that accepts one argument (string) and returns a modified version of that string.

*Details:* Preprocess the prefix with a custom function

*Examples:*

```
err <- Logger$new("INFO")$hook(crayon::red)
err$log("hello")
```

**Method** `dir()`:

*Usage:*

```
Logger$dir()
```

*Details:* Include the directory in the log

*Examples:*

```
info <- Logger$new("INFO")$dir()
info$log("directory")
```

### Method `flag()`:

*Usage:*

```
Logger$flag(what)
```

*Arguments:*

what Function to run for every message logged or string to include in log message.

*Details:* Pass a custom flag

*Examples:*

```
fl <- function(){
 paste0(sample(letters, 4), collapse = "")
}

info <- Logger$new("INFO")$flag(fl)
info$log("random")
```

### Method `log()`:

*Usage:*

```
Logger$log(..., sep = " ", collapse = " ")
```

*Arguments:*

... Elements to compose message.

sep, collapse Separators passed to [paste()](paste()).

*Details:* Log messages

*Examples:*

```
info <- Logger$new("INFO")
info$log("Logger")
```

### Method `dump()`:

*Usage:*

```
Logger$dump(file = "dump.log")
```

*Arguments:*

file Name of the file to dump the logs to.

*Details:* Dump the log to a file

*Examples:*

```
info <- Logger$new("INFO")
info$log("hello")
\dontrun{info$dump()}
```

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

*Usage:*

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

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
info <- Logger$new("INFO")$
 date()$
 time()$
 hook(crayon::blue)

info$log("Hello")
Sys.sleep(.7)
info$log("World")


## ------------------------------------------------
## Method `Logger$new`
## ------------------------------------------------

info <- Logger$new("INFO")
info$log("hello")

## ------------------------------------------------
## Method `Logger$date`
## ------------------------------------------------

info <- Logger$new("INFO")$date()
info$log("today")

## ------------------------------------------------
## Method `Logger$time`
## ------------------------------------------------

info <- Logger$new("INFO")$time()
info$log("now")

## ------------------------------------------------
## Method `Logger$unix`
## ------------------------------------------------

info <- Logger$new("INFO")$unix()
info$log("timestamp")

## ------------------------------------------------
## Method `Logger$hook`
## ------------------------------------------------

err <- Logger$new("INFO")$hook(crayon::red)
err$log("hello")

## ------------------------------------------------
## Method `Logger$dir`
## ------------------------------------------------

info <- Logger$new("INFO")$dir()
info$log("directory")
```

```
## ------------------------------------------------
## Method `Logger$flag`
## ------------------------------------------------

fl <- function(){
 paste0(sample(letters, 4), collapse = "")
}

info <- Logger$new("INFO")$flag(fl)
info$log("random")

## ------------------------------------------------
## Method `Logger$log`
## ------------------------------------------------

info <- Logger$new("INFO")
info$log("Logger")

## ------------------------------------------------
## Method `Logger$dump`
## ------------------------------------------------

info <- Logger$new("INFO")
info$log("hello")
## Not run: info$dump()
```

| plumber | *Plumber* |
|---------|-----------|

### Description

Serve a plumber API with metrics, use this function like you would use plumber::pr().

### Usage

```
prLog(file = NULL, ..., requests = requestLogger)

prWithLog(pr, requests = requestLogger)
```

### Arguments

| | |
|---|---|
| file | Plumber file as passed to plumber::pr(). |
| ... | Any other argument to pass to plumber::pr(). |
| requests | Logger to log requests, set to NULL to not log. |
| pr | Plumber API as returned by plumber::pr(). |

---

shiny                                    *Default Logger for Shiny.*

---

### Description

Serve a shiny application with default loggers, useful for debugging.

### Usage

```
logApp(ui, server, ..., inputs = inputLogger)

shinyWithLog(app, inputs = inputLogger)
```

### Arguments

| | |
|---|---|
| ui, server | UI definition and server function as passed to shiny::shinyApp(). |
| ... | Any other arguments passed to shiny::shinyApp(). |
| inputs | Logger to log inputs, set to NULL to not log. |
| app | Shiny application as returned by shiny::shinyApp(). |

---

templates                                *Template Loggers*

---

### Description

Template loggers for convenience.

### Usage

```
infoLog(prefix = "INFO")

errorLog(prefix = "ERROR")

warningLog(prefix = "WARNING")

successLog(prefix = "SUCCESS")
```

### Arguments

| | |
|---|---|
| prefix | The prefix to use, this is passed to Logger. |

### Examples

```
info <- infoLog()
info$log("Information")
```

# Index