Package: dyn.log (via r-universe)

September 10, 2024

Type Package

Title Dynamic Logging for R Inspired by Configuration Driven Development

Version 0.4.1.9000

Maintainer Brandon Moretz <bmoretz@ionicsolutions.net>

Description A comprehensive and dynamic configuration driven logging package for R. While there are several excellent logging solutions already in the R ecosystem, I always feel constrained in some way by each of them. Every project is designed differently to solve it's domain specific problem, and ultimately the utility of a logging solution is its ability to adapt to this design. This is the raison d'être for 'dyn.log': to provide a modular design, template mechanics and a configuration-based integration model, so that the logger can integrate deeply into your design, even though it knows nothing about it.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

Imports crayon (>= 1.4.1), glue (>= 1.4.2), R6 (>= 2.5.1), rlang (>= 0.4.12), stringr (>= 1.4.0), yaml (>= 2.2.1)

Roxygen list(markdown = TRUE)

URL https://bmoretz.github.io/dyn.log/

BugReports https://github.com/bmoretz/dyn.log/issues

Suggests devtools, usethis, rmarkdown, markdown, knitr, covr (>= 3.5.1), testthat (>= 3.0.0), lintr (>= 0.28), remotes, pkgdown, prettydoc, here, fansi, pander, DT

Config/testthat/edition 3

RoxygenNote 7.1.2

Language en-US

VignetteBuilder knitr

Repository https://bmoretz.r-universe.dev

RemoteUrl https://github.com/bmoretz/dyn.log

RemoteRef HEAD

RemoteSha d5f7548e1e96c372b47505617ce4d59341016c7e

Contents

`.onAttach` *Attach Handler Package initialization routine.*

Description

Attach Handler

Package initialization routine.

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	library name
`pkgname`	package name

.onDetach *Detach Handler Package initialization routine.*

Description

Detach Handler

Package initialization routine.

Usage

```
.onDetach(libpath)
```

Arguments

libpath	libpath

.onLoad *Load Handler Package initialization routine.*

Description

Load Handler

Package initialization routine.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

libname	library name
pkgname	package name

active *Active Settings*

Description

Package environment variable to hold global level settings.

Usage

```
active
```

Format

An object of class environment of length 0.

apply_active_settings *Apply Active Logger Settings*

Description

Parses and loads the settings specified in the logger configuration and ensures they are active in the environment.

Usage

```
apply_active_settings(settings)
```

Arguments

settings	defined in the configuration

See Also

Other Configuration: config_specification(), create_log_levels(), display_log_levels(), ensure_logger(), get_active_settings(), load_log_layouts(), wipe_logger()

as.character.log_level *Get Log Level Name*

Description

gets the name of the log level though casting to a character and forwarding the call to get_level_name.

Usage

```
## S3 method for class 'log_level'
as.character(x, ...)
```

Arguments

x	log level
...	ignored

Value

log level name

as.integer.log_level *Gets the severity of a log level.*

Description

Gets the severity of a log level.

Usage

```
## S3 method for class 'log_level'
as.integer(x, ...)
```

Arguments

x	log level
...	ignored

Value

log level

class_scope *Calling Class Scope*

Description

Gets the exposed public field scope of a R6 class. Used for evaluating cls field execution scopes.

Usage

```
class_scope(cls)
```

Arguments

cls	R6 class to export.

Value

system context for evaluating `fmt_metric` objects.

See Also

Other Context: `exec_context()`, `format_fn_call()`, `get_call_stack()`, `get_r_version()`, `get_system_info()`, `sys_context()`

clean_internal_calls *Clean System Calls*

Description

Cleans up any internal system calls from inside the package from the call stack.

Usage

```
clean_internal_calls(call_stack)
```

Arguments

call_stack	call stack

Value

string representation of a func call.

See Also

Other Internal: extract_func_name(), is_logger_call()

config_specification *Config Specification*

Description

Loads & attaches a logger with the specified config.

Usage

```
config_specification()
```

Value

Nothing.

See Also

Other Configuration: apply_active_settings(), create_log_levels(), display_log_levels(), ensure_logger(), get_active_settings(), load_log_layouts(), wipe_logger()

create_log_levels *Attach Log Levels*

Description

Parses and loads the levels specified in the logging configuration and registers them with the dispatcher via the `log_levels` active binding.

Usage

```
create_log_levels(definitions)
```

Arguments

`definitions`	defined in the configuration

See Also

Other Configuration: `apply_active_settings()`, `config_specification()`, `display_log_levels()`, `ensure_logger()`, `get_active_settings()`, `load_log_layouts()`, `wipe_logger()`

display_log_levels *Display Log Levels*

Description

A utility function that dynamically displays the configured log levels (loaded from config), and outputs them in a simple layout with only the log level and msg formatted in their crayon styles.

Usage

```
display_log_levels()
```

See Also

Other Configuration: `apply_active_settings()`, `config_specification()`, `create_log_levels()`, `ensure_logger()`, `get_active_settings()`, `load_log_layouts()`, `wipe_logger()`

ensure_logger *Ensure Instance*

Description

Ensures there is an active dispatcher attached to the specified environment.

Usage

```
ensure_logger(variable)
```

Arguments

variable	variable name.

Value

None.

See Also

Other Configuration: apply_active_settings(), config_specification(), create_log_levels(), display_log_levels(), get_active_settings(), load_log_layouts(), wipe_logger()

evaluate_layout *Evaluate Layout*

Description

Evaluates a log layout, which is simply a container for a set of formats that specify the log entry layout.

Usage

```
evaluate_layout(detail, context)
```

Arguments

detail	The details of the layout specified for evaluation.
context	a list of contexts needed to evaluate formats in the the layout.

Value

evaluated log layout

See Also

Other Log Layout: new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp(), new_log_layout()

---

exec_context *Execution Context*

Description

Wrapper around Sys.info() and get_r_version that provides a consolidated list of variables used for logging contexts.

Usage

```
exec_context(
  keep_args = FALSE,
  max_calls = 5,
  call_subset = c(-1, -1),
  filter_internal = TRUE
)
```

Arguments

keep_args	bool to specify keep function all arguments
max_calls	maximum number of calls to keep from the stack
call_subset	offset index into system calls
filter_internal	filter out internal calls?

Value

system context for evaluating fmt_metric objects.

See Also

Other Context: class_scope(), format_fn_call(), get_call_stack(), get_r_version(), get_system_info(), sys_context()

extract_func_name *Extract Function Name*

Description

Extracts the name of the function from a deparse call.

Usage

```
extract_func_name(func)
```

Arguments

func	function name

Value

function name without arguments

See Also

Other Internal: clean_internal_calls(), is_logger_call()

format.fmt_timestamp *Gets the format of a format object.*

Description

Gets the format of a format object.

Usage

```
## S3 method for class 'fmt_timestamp'
format(x, ...)
```

Arguments

x	object to extract value from.
...	further arguments passed to or from other methods.

Value

object's value

format.log_level *Log Level Format*

Description

formats a message with the style of the log level.

Usage

```
## S3 method for class 'log_level'
format(x, message = character(0), ...)
```

Arguments

x	log level
message	message to format
...	further arguments passed to or from other methods.

Value

styled level information

Examples

```
## Not run:
level_info(LEVEL)

## End(Not run)
```

format_fn_call *Format Function Call*

Description

Formats a function call into a deparsed string.

Usage

```
format_fn_call(expr, cutoff = 100L)
```

Arguments

expr	function call
cutoff	max width cutoff

Value

string representation of a func call.

See Also

Other Context: class_scope(), exec_context(), get_call_stack(), get_r_version(), get_system_info(), sys_context()

---

get_active_settings *Active Logger Settings*

Description

Gets the active global settings for the logger.

Usage

```
get_active_settings()
```

See Also

Other Configuration: apply_active_settings(), config_specification(), create_log_levels(), display_log_levels(), ensure_logger(), load_log_layouts(), wipe_logger()

---

get_call_stack *Formatted Call Stack*

Description

Placeholder for the formatted call stack in a log layout.

Usage

```
get_call_stack(
  keep_args = FALSE,
  call_subset = c(-1, -1),
  filter_internal = TRUE
)
```

Arguments

keep_args	T/F to indicate if you want to keep call arguments or not.
call_subset	offset index into system calls
filter_internal	filter out internal calls?

Value

formatted call stack

See Also

Other Context: class_scope(), exec_context(), format_fn_call(), get_r_version(), get_system_info(), sys_context()

---

get_configurations *Get Configurations*

Description

Gets all available logging configurations exposed by the package.

Usage

```
get_configurations(pkgname = "dyn.log")
```

Arguments

pkgname	package name to get configs for.

See Also

Other Logging: LogDispatch, init_logger()

---

get_r_version *R Version*

Description

Wrapper around R.Version() to produce a nicely formatted string for use use in sys_context.

Usage

```
get_r_version()
```

Value

R environment version is (major).(minor) format

See Also

Other Context: class_scope(), exec_context(), format_fn_call(), get_call_stack(), get_system_info(), sys_context()

get_system_info *Get System Info*

Description

Wrapper around Sys.info() that provides the values in a named list format.

Usage

```
get_system_info()
```

Value

`Sys.info()` as a named list

See Also

Other Context: `class_scope()`, `exec_context()`, `format_fn_call()`, `get_call_stack()`, `get_r_version()`, `sys_context()`

init_logger *Init Logger*

Description

Loads the configuration passed in, or uses the default if nothing is specified, and attaches a reference to the singleton dispatcher to the global environment.

Usage

```
init_logger(file_path = NULL)
```

Arguments

`file_path`	logging configuration to use.

See Also

Other Logging: `LogDispatch`, `get_configurations()`

is_logger_call *Is Logger Call*

Description

Determines if a call came from the logger, so we can exclude it from the call stack.

Usage

```
is_logger_call(call)
```

Arguments

call	function call

Value

string representation of a func call.

See Also

Other Internal: clean_internal_calls(), extract_func_name()

length.log_layout *Log Layout Length*

Description

Generic override for length of a log layout that returns the number of individual format objects in the layout.

Usage

```
## S3 method for class 'log_layout'
length(x, ...)
```

Arguments

x	log format
...	further arguments passed to or from other methods.

Value

number of formats in the layout.

level_description *Log Level Description*

Description

Gets the description of a log level.

Gets the description of a log level.

Usage

```
level_description(level)

level_description(level)
```

Arguments

level	log level

Value

level description

level severity

Examples

```
## Not run:
level_description(LEVEL)

## End(Not run)
## Not run:
level_description(LEVEL)

## End(Not run)
```

level_info *Log Level Information*

Description

Gets log level information.

Usage

```
level_info(level)
```

Arguments

level	log level

Value

log level information

Examples

```
## Not run:
level_info(LEVEL)

## End(Not run)
```

---

level_name *Get Level Name*

Description

gets the name of the log level.

Usage

```
level_name(level)
```

Arguments

level	log level

Value

log level name

---

level_severities *Level Severities*

Description

Gets the severity associated with each log level.

Usage

```
level_severities()
```

Value

styled level information

level_severity *get level severity*

Description

Gets the severity of a log level.

Usage

```
level_severity(level)

level_severity(level)
```

Arguments

level	log level

Value

level severity

level severity

Examples

```
## Not run:
level_severity(LEVEL)

## End(Not run)
## Not run:
level_severity(LEVEL)

## End(Not run)
```

load_log_layouts *Load Log Layouts*

Description

Parses and loads layouts specified in the logging configuration and registers them with the log dispatcher via the `log_layouts` active binding.

Usage

```
load_log_layouts(layouts)
```

Arguments

layouts	defined in the configuration

Value

None.

See Also

Other Configuration: apply_active_settings(), config_specification(), create_log_levels(), display_log_levels(), ensure_logger(), get_active_settings(), wipe_logger()

LogDispatch *Log Dispatch*

Description

R6 Class that dispatches log messages throughout the application.

Details

This object is designed to a centralized logging dispatcher that renders log messages with the appropriate context of the calling object. The log_layout() object is used to generate log message layouts (render formats), which are used by the LogDispatcher to render highly-customizable and detailed log messages.

Methods

Public methods:

- LogDispatch$new()
- LogDispatch$attach_log_level()

Method new(): Creates a new instance of a log config.

Usage:

```
LogDispatch$new()
```

Returns: A new LogLayout object.

Method attach_log_level(): Attaches a S3 log_level object to the log dispatcher by creating a new function wrapping the specified log level, and binding and instance of the dispatcher quote block with the context of the log level.

Usage:

```
LogDispatch$attach_log_level(log_level)
```

Arguments:

log_level	log level to attach

See Also

Other Logging: get_configurations(), init_logger()

log_layouts *Log Layouts*

Description

an active binding to keep track of log layouts created with `new_log_layout`.

Usage

```
log_layouts(association = character(0), layout = NULL)
```

Arguments

`association`	named association to the layout
`layout`	log layout to add if not already existing.

Value

defined log layouts

log_layout_detail *Log Layout Detail*

Description

Gets the layout formats and the distinct format types in a log layout instance, which is useful for determining the appropriate amount of log context to construct.

Usage

```
log_layout_detail(layout)
```

Arguments

`layout`	object to extract layout detail from.

Value

layout format

log_levels *Log Levels*

Description

an active binding to keep track of log levels created with `new_log_level`.

Usage

```
log_levels(name = character(0), level = NULL)
```

Arguments

`name`	name associated with the log level
`level`	log level to add if not already existing.

Value

defined log levels

new_fmt_cls_field *Formatted field from the calling class scope.*

Description

Placeholder for a container class field

Usage

```
new_fmt_cls_field(style, field)
```

Arguments

`style`	crayon::style()
`field`	field in the object to display

Value

`new_fmt_cls_field`

See Also

Other Log Layout: `evaluate_layout()`, `new_fmt_exec_scope()`, `new_fmt_layout()`, `new_fmt_line_break()`, `new_fmt_literal()`, `new_fmt_log_msg()`, `new_fmt_metric()`, `new_fmt_timestamp()`, `new_log_layout()`

new_fmt_exec_scope *Formatted variable from the execution scope.*

Description

Placeholder for an execution scope variable.

Usage

```
new_fmt_exec_scope(style, field)
```

Arguments

style	crayon::style()
field	execution scope field

Value

new_fmt_cls_field

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_layout(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp(), new_log_layout()

new_fmt_layout *Format Layout*

Description

Base type for log format objects.

Usage

```
new_fmt_layout(style)
```

Arguments

style	crayon that the layout will use in log generation.

Value

new log format

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp(), new_log_layout()

new_fmt_line_break *Formatted Line Break*

Description

Placeholder for a new line in a log layout.

Usage

```
new_fmt_line_break()
```

Value

log layout newline.

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp(), new_log_layout()

new_fmt_literal *Formatted Literal*

Description

Placeholder for a formatted literal in a log layout.

Usage

```
new_fmt_literal(style, literal)
```

Arguments

style	format style (crayon)
literal	log value

Value

log metric layout.

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_line_break(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp(), new_log_layout()

Examples

```
## Not run:
new_fmt_literal(red $ bold, "literal text")

new_fmt_literal(blue $ italic, "literal text")

## End(Not run)
```

`new_fmt_log_level` *Formatted Log Level*

Description

Placeholder for the formatted log level in a log layout.

Usage

```
new_fmt_log_level()
```

Value

a `fmt_log_level`.

`new_fmt_log_msg` *Formatted Messaged, based on log level*

Description

Placeholder for the log msg in a log layout.

Usage

```
new_fmt_log_msg()
```

Value

log layout newline.

See Also

Other Log Layout: `evaluate_layout()`, `new_fmt_cls_field()`, `new_fmt_exec_scope()`, `new_fmt_layout()`, `new_fmt_line_break()`, `new_fmt_literal()`, `new_fmt_metric()`, `new_fmt_timestamp()`, `new_log_layout()`

new_fmt_metric *Formatted Metric*

Description

Inserts a formatted log metric.

Usage

```
new_fmt_metric(style, metric)
```

Arguments

style	that the layout will use in log generation
metric	the metric to log.

Value

a new formatted metric

See Also

LogDispatch

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_timestamp(), new_log_layout()

Examples

```
## Not run:
new_fmt_metric(bold $ green, "sysname")

new_fmt_metric(bold $ red, "release")

## End(Not run)
```

new_fmt_timestamp *Formatted Time stamp*

Description

Placeholder for a formatted time stamp in a log layout.

Usage

```
new_fmt_timestamp(style, format = "[%x %H:%M:%S %z]")
```

Arguments

style	format style (crayon)
format	time stamp format, defaults to: %x %H:%M:%S %z, e.g., 12/04/21 14:31:25 -0500

Value

log metric layout.

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_log_layout()

Examples

```
## Not run:
fmt_timestamp(red $ bold, "%Y-%m-%d %H:%M:%S")

fmt_timestamp(blue $ italic, "%x %H:%M:%S %z")

## End(Not run)
```

---

new_log_layout *Log Layout*

Description

a class that stores a collection of log format objects and understands how to associate a given format to a class of objects.

Usage

```
new_log_layout(
  format = list(),
  seperator = " ",
  new_line = "\n",
  association = character()
)
```

Arguments

format	collection of format objects to initialize with.
seperator	format entry separator, defaults to a single space.
new_line	the layout separator that is inserted between lines.
association	objects to associate this log format with.

Value

object's value

See Also

Other Log Layout: evaluate_layout(), new_fmt_cls_field(), new_fmt_exec_scope(), new_fmt_layout(), new_fmt_line_break(), new_fmt_literal(), new_fmt_log_msg(), new_fmt_metric(), new_fmt_timestamp()

---

new_log_level *Log Level*

Description

S3 object to represent a typed & predefined log level.

Usage

```
new_log_level(name, description, severity, log_style = NULL, msg_style = NULL)
```

Arguments

name	name of the log level is the string representation.
description	description of the log level & limited info on appropriate usage.
severity	log severity is used in determining if a message should get displayed according to the currently set evaluation threshold.
log_style	is a crayon::style() that will colorize the log level.
msg_style	is a crayon::style() style that will gray scale the log message, with typically inverted strength, according to the severity.

Value

log_level

style *Style*

Description

Generic style method used for overriding to get style information from various logging objects.

Usage

```
style(obj)
```

Arguments

obj	object to extract value from.

Value

object's value

style.fmt_layout *Style*

Description

Gets the style of a format object.

Usage

```
## S3 method for class 'fmt_layout'
style(obj, ...)
```

Arguments

obj	object to extract value from.
...	further arguments passed to or from other methods.

Value

object's value

style.log_level *Get Level Style*

Description

gets the style of the log level.

Usage

```
## S3 method for class 'log_level'
style(obj, ...)
```

Arguments

obj	log level
...	further arguments passed to or from other methods.

Value

log level name

sys_context *System Context*

Description

Wrapper around Sys.info() and get_r_version that provides a consolidated list of variables used for logging contexts.

Usage

```
sys_context()
```

Value

system context for evaluating fmt_metric objects.

Metrics

System Context

- "sysname" : The operating system name.
- "release" : The OS release.
- "version" : The OS version.
- "nodename" : A name by which the machine is known on the network (if any).

- "machine" : A concise description of the hardware, often the CPU type.
- "login" : The user's login name, or "unknown" if it cannot be ascertained.
- "user" : The name of the real user ID, or "unknown" if it cannot be ascertained.
- "r-ver" : R Version in (major).(minor) format.

See Also

Other Context: `class_scope()`, `exec_context()`, `format_fn_call()`, `get_call_stack()`, `get_r_version()`, `get_system_info()`

`value` *Value*

Description

Base method for getting the value of a format object.

Usage

```
value(obj, ...)
```

Arguments

`obj`	object to extract value from.
`...`	further arguments passed to or from other methods.

Value

object's value

`value.fmt_cls_field` *Value*

Description

Generic override for getting the value of an enclosing class variable.

Usage

```
## S3 method for class 'fmt_cls_field'
value(obj, cls_context, ...)
```

Arguments

obj	object to extract value from.
cls_context	class scope to evaluate with.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_exec_scope *Value*

Description

Generic override for getting the value of an execution scope variable.

Usage

```
## S3 method for class 'fmt_exec_scope'
value(obj, env_context, ...)
```

Arguments

obj	object to extract value from.
env_context	class scope to evaluate with.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_literal *Value*

Description

Generic override for getting the value of a literal log message.

Usage

```
## S3 method for class 'fmt_literal'
value(obj, ...)
```

Arguments

obj	object to extract value from.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_log_level *Value*

Description

Generic override for getting the value for log level information.

Usage

```
## S3 method for class 'fmt_log_level'
value(obj, lvl_context, ...)
```

Arguments

obj	object to extract value from.
lvl_context	context to evaluate log level.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_log_msg *Value*

Description

Generic override for getting the value of an log format message.

Usage

```
## S3 method for class 'fmt_log_msg'
value(obj, msg_context, ...)
```

Arguments

obj	object to extract value from.
msg_context	context to evaluate log message.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_metric *Value*

Description

Generic override for getting the value of an system info variable.

Usage

```
## S3 method for class 'fmt_metric'
value(obj, sys_context, ...)
```

Arguments

obj	object to extract value from.
sys_context	context to evaluate the metric.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_newline *Value*

Description

Generic override for getting the value of a new line placeholder.

Usage

```
## S3 method for class 'fmt_newline'
value(obj, ...)
```

Arguments

obj	object to extract value from.
...	further arguments passed to or from other methods.

Value

object's value

---

value.fmt_timestamp *Value*

Description

Generic override for getting the value of a formatted timestamp.

Usage

```
## S3 method for class 'fmt_timestamp'
value(obj, ...)
```

Arguments

obj	object to extract value from.
...	further arguments passed to or from other methods.

Value

object's value

---

wipe_logger *Wipe the Logger Instance*

Description

Cleans up any dangling global instance from a previous load.

Usage

```
wipe_logger()
```

Value

None.

See Also

Other Configuration: apply_active_settings(), config_specification(), create_log_levels(), display_log_levels(), ensure_logger(), get_active_settings(), load_log_layouts()

Index