Levels

Brandon Moretz

2021-12-27

Dynamic Log Levels

```
library(dyn.log)
```

Definitions

One of the main objectives of *dyn.log* is to be as configuration driven as possible, which allows the logger to conform to the problem - not the other way around. Log levels are fully configuration driven, and they are specified in the logger configuration with the following schema:

```
levels:
- name: DEBUG
  description: This level designates fine-grained informational events that are
        most useful to debug an application.
  severity: !expr 500L
  log_style: !expr crayon::make_style('deepskyblue2')$bold
  msg_style: !expr crayon::make_style('gray90')
```

The levels node contains the log levels you want available in your environment. When a log level is defined in the configuration, it automatically becomes accessible via a first-class function on the dispatcher, e.g.:

Attributes

The attributes that define a log level are:

- **name**: what gets displayed as the level when a log message is rendered.
- **description**: a brief 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**: a crayon that will colorize the log level.
- **msg_style**: a crayon style that will gray scale the log message, with typically inverted strength, according to the severity.

Invocation

Once you have defined a log level in the logger configuration, the dispatcher will automatically have a method corresponding to that level. This enables a fairly intuitive approach to invoking the log dispatcher, e.g., if you have the previous *debug* level defined in your configuration this is how you would log a message with that level:

```
var1 <- "abc"; var2 <- 123; var3 <- round(runif(1), digits = 6)

Logger$debug("my log message - var1: {var1}, var2: {var2}, var3: {var3}")
```

```
DEBUG [12/27/21 01:05:20 -0500] my log message - var1: abc, var2: 123, var3: 0.183161
```

You'll notice that all log messages by default use the standardize glue format so local variables are capturable in the log output.

Localization

You can see what log levels have been specified in your logging configuration by calling *log_levels()*.

```
log_levels()
[1] "trace"   "debug"   "info"    "success" "warn"    "error"   "fatal"
```

Which will output only the *names* of the defined levels, not all of the detail that defines them.

Detail

To get the details about a defined level you can call *level_info()*:

```
level_info("debug")
```

```
$name
[1] "DEBUG"

$description
[1] "This level designates fine-grained informational events that are most useful to d

$severity
[1] 500

$style
$style$level
Crayon style function, deepskyblue2, bold: example output.

$style$message
```

```
Crayon style function, gray90: example output.

$style$example
DEBUG - This level designates fine-grained informational events that are most useful t
```

The detailed information about a log level shows every configurable attribute about the level, and an example of how the level renders with its associated crayon styles for the level and message. A vanilla log layout consisting of only the level and msg would get rendered with the look of the example attribute.

Configuration

Out of the Box

The default (OTB) logging configuration closely resembles the fairly ubiquitous log4j level scheme.

There is a utility method called *display_log_levels()* that will output all loaded log levels from the configuration rendered with their defined styles & definitions:

```
display_log_levels()
```

```
TRACE This level designates finer-grained informational events than the DEBUG.

DEBUG This level designates fine-grained informational events that are most useful to

INFO This level designates informational messages that highlight the progress of the a

SUCCESS This level designates that the operation was unencumbered.

WARN This level designates potentially harmful situations.

ERROR This level designates error events that might still allow the application to con

FATAL This level designates very severe error events that will presumably lead the app
```

Note: even through fansi is amazing at rendering these in html format, your terminal may look slightly different; expecialy the grayscale colors on log messages.

Customizing

To customize the log levels in your environment, please see the configuration vignette.