CustomFormatters

How to write your own custom formatters.

Introduction

Formatters may optionally be added to any logger. The most common example is a formatter which prepends the timestamp to log messages being written to a file. To achieve this you may simple create and add a formatter to the existing file logger. Using formatters gives you the ability to customize the log message appearance without having to rewrite the entire logger.

Formatters can also be used to filter log messages. The criteria for filtering is entirely up to you.

And remember that formatters are applied individually to loggers. So you can format and/or filter on a per-logger basis.

Details

It is incredibly simple to create your own custom formatter. The protocol for DDLogFormatter is defined in DDLog.h:

```
@protocol DDLogFormatter <NSObject>
@required

/**
 * Formatters may optionally be added to any logger.
 * This allows for increased flexibility in the logging environment.
 * For example, log messages for log files may be formatted differently than log messages for the console.
 *
 * The formatter may also optionally filter the log message by returning nil.
**/

- (NSString *)formatLogMessage:(DDLogMessage *)logMessage;

@end
```

It's pretty straight-forward. You just need to implement a single method.

The DDLogMessage object contains the information about the log message including:

- LogLevel - log level of message
- LogMsg - original log message
- File - name of the file the log message came from
- Function - method the log message came from
- LineNumber - line number is file where the log message came from
- Timestamp - when the log message was executed
- ThreadID - which thread issued the log message

Let's write a simple formatter that automatically prepends the log level and timestamp before every log message:

MyCustomFormatter.h

```
#import <Foundation/Foundation.h>
#import "DDLog.h"

@interface MyCustomFormatter : NSObject <DDLogFormatter>
{
    NSDateFormatter *dateFormatter;
}

@end
```

MyCustomFormatter.m

```
#import "MyCustomFormatter.h"

@implementation MyCustomFormatter

- (id)init
{
    if((self = [super init]))
    {
        dateFormatter = [[NSDateFormatter alloc] init];
        [dateFormatter setFormatterBehavior:NSDateFormatterBehavior10_4];
        [dateFormatter setDateFormat:@"yyyy/MM/dd HH:mm:ss:SSS"];
        [dateFormatter setDateFormat:@"yyyy/MM/dd HH:mm:ss:SSS"];
    }
    return self;
}

- (NSString *)formatLogMessage:(DDLogMessage *)logMessage
{
        NSString *logLevel;
        switch (logMessage->logLevel)
        {
                case LOG_LEVEL_ERROR : logLevel = @"E"; break;
                case LOG_LEVEL_WARN  : logLevel = @"W"; break;
                case LOG_LEVEL_INFO  : logLevel = @"I"; break;
                default              : logLevel = @"V"; break;
        }

        NSString *dateAndTime = [dateFormatter stringFromDate:(logMessage->timestamp)];

        NSString *logMsg = logMessage->logMsg;

        return [NSString stringWithFormat:@"%@ %@ | %@\n", logLevel, dateAndTime, logMsg];
}

- (void)dealloc
{
    [dateFormatter release];
    [super dealloc];
}

@end
```

The idea is to get log messages like this:

```
DDLogError(@"Paper Jam!");       // E 2010/05/20 15:33:18:621 | Paper Jam!
DDLogWarn(@"Low toner.");        // W 2010/05/20 15:33:18:621 | Low toner.
DDLogInfo(@"Doc printed.");      // I 2010/05/20 15:33:18:621 | Doc printed.
DDLogVerbose(@"Init doc_parse"); // V 2010/05/20 15:33:18:621 | Init doc_parse.
```