(/)

How to Log to the Console in Color

Last modified: March 8, 2023

Written by: Shaun Phillips
(https://www.baeldung.com/author/shaunphillips)

Logging (https://www.baeldung.com/category/logging)

Debugging (https://www.baeldung.com/tag/debugging)

1. Introduction

Adding some color can make logging much easier to read.

In this article, we'll see how to add color to our logs for consoles such as the Visual Studio Code terminal, Linux, and Windows command prompt.

Before we start, let's note that, unfortunately, there are only limited color settings in the Eclipse IDE console. The console within Eclipse IDE does not support color determined by Java code, so **the solutions presented in this article will not work within the Eclipse IDE console.**

2. How to Use ANSI Codes to Color Logs

The easiest way to achieve colorful logging is by using ANSI escape sequences, (https://en.wikipedia.org/wiki/ANSI_escape_code) often referred to as ANSI codes.

ANSI codes are special sequences of bytes that some terminals interpret as a command.


```
System.out.println("Here's some text");
System.out.println("\u001B[31m" + "and now the text is red");
```


In the output, we see that the ANSI code was not printed, and the color of the font has changed to red:

```
Here's some text
and now the text is red
```

Let's note that we need to make sure we reset the font color once we're done logging.

Fortunately, this is easy. We can simply print *\u001B[31m*, which is the ANSI reset command.

The reset command will reset the console to its default color. Note that this may not necessarily be black, it could be white, or any other color configured by the console. For example:

```
System.out.println("Here's some text");
System.out.println("\u001B[31m" + "and now the text is red" +
"\u001B[0m");
System.out.println("and now back to the default");
```

Gives the output:

```
Here's some text
and now the text is red
and now back to the default
```

Most logging libraries will obey ANSI codes, which allows us to build some colorful loggers.

For example, we could quickly construct a logger that uses different colors for different log levels.

```
public class ColorLogger {

    private static final Logger LOGGER =
LoggerFactory.getLogger(ColorLogger.class);

    public void logDebug(String logging) {
        LOGGER.debug("\u001B[34m" + logging + "\u001B[0m");
    }
    public void logInfo(String logging) {
        LOGGER.info("\u001B[32m" + logging + "\u001B[0m");
    }

    public void logError(String logging) {
        LOGGER.error("\u001B[31m" + logging + "\u001B[0m");
    }
}
```

Let's use this to print some color to the console:

```
ColorLogger colorLogger = new ColorLogger();
colorLogger.logDebug("Some debug logging");
colorLogger.logInfo("Some info logging");
colorLogger.logError("Some error logging");
```

```
[main] DEBUG com.baeldung.color.ColorLogger - Some debug
logging
[main] INFO com.baeldung.color.ColorLogger - Some info logging
[main] ERROR com.baeldung.color.ColorLogger - Some error
logging
```

We can see that each log level is a different color, making our logs much more readable.

Finally, ANSI codes can be used to control much more than just font color – we can control background colors and font-weight, and style. There's a selection of these ANSI codes in the example project.

3. How to Color Logs in the Windows Command Prompt

Unfortunately, some terminals don't support ANSI codes. One prime example is the Windows command prompt, and the above won't work. Therefore, we need a more sophisticated solution.

However, rather than trying to implement it ourselves, we can leverage an established library called JANSI (https://github.com/fusesource/jansi) to our pom.xml:

```
<dependency>
    <groupId>org.fusesource.jansi</groupId>
    <artifactId>jansi</artifactId>
    <version>2.4.0</version>
</dependency>
```

Now to log in color, we can simply call into the ANSI API that JANSI provides:

```
private static void logColorUsingJANSI() {
    AnsiConsole.systemInstall();

    System.out.println(ansi()
        .fgRed()
        .a("Some red text")
        .fgBlue()
        .a(" and some blue text")
        .reset());

    AnsiConsole.systemUninstall();
}
```

This produces the text:

```
Some red text and some blue text
```

Let's note that **we have to first install the *AnsiConsole*, and uninstall it once we're done.**

As with ANSI codes, JANSI also provides a large range of logging formats.

JANSI achieves this functionality by detecting the terminal being used and invoking the appropriate platform-specific API required. This means that when JANSI detects a Windows Command Prompt, rather than use ANSI codes that don't work, it invokes libraries that use Java Native Interface (JNI) (https://baeldung.com/java-native) methods.

Finally, JANSI will also ensure it strips out unwanted ANSI codes when the environment doesn't require them, helping to keep our logs clean and tidy.

Overall, JANSI provides us with a powerful and convenient way to log in color to most environments and terminals.

4. Conclusion

In this article, we learned how to use ANSI codes to control the color of the console font and saw an example of how we could distinguish log levels using color.

Finally, we found that not all consoles support ANSI codes and highlighted one such library, JANSI, that provides more sophisticated support.

As always, the example project is available over on GitHub (https://github.com/eugenp/tutorials/tree/master/core-java-modules/core-java-console).

Comments are closed on this article!

COURSES

ALL COURSES (/ALL-COURSES)
ALL BULK COURSES (/ALL-BULK-COURSES)
ALL BULK TEAM COURSES (/ALL-BULK-TEAM-COURSES)
THE COURSES PLATFORM (HTTPS://COURSES.BAELDUNG.COM)

SERIES

JAVA "BACK TO BASICS" TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
APACHE HTTPCLIENT TUTORIAL (/HTTPCLIENT-GUIDE)
REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)
SPRING REACTIVE TUTORIALS (/SPRING-REACTIVE-GUIDE)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (/FULL_ARCHIVE)
EDITORS (/EDITORS)
JOBS (/TAG/ACTIVE-JOB/)