Proposal

Evan Lee

April 2020

1 Problem Statement

The problem I aim to solve is two-fold: (a) remove some of the human component around logging usage decisions and (b) remove some of the boilerplate around standard logging use cases. These two parts of the problem are tightly related, as a lot of identified logging use cases require a lot of boilerplate – however, due to this, there is often a human component of deciding how much or little to implement. My research will abstract away these components and instead provide a standardized logging library to both eliminate developer overhead in certain logging use cases.

A great example of the problem I want to tackle is python-zeep/transports.py. As you can see in the linked method is 29 lines long where 21 of those lines deal in one way or another with logging or making decisions on log statements. Realistically, the functionality of the given method could be collapsed into one list – the proxy call to the requests module on line 61. However, logging has driven an explosion of code in this particular case.

2 Implementation

I intend to solve this problem by developing a logging library for python and implementing my research topics in the library. I also intend to do additional research into some major logging patterns in open-source and identifying one or more major use-cases that I want to chase down abstracting and implementing into my logging library.

3 Design

I intend to base my solution similarly to my original paper of interest ("Log++: Logging for a Cloud-Native World"), using a similar structure but tackling a slightly different problem. Marron implements a logging library for Node.js applications in order to tackle performance issues that arise with logging. However, I intend to implement a logging library to abstract a boilerplate away from an as-yet-unidentified use case.

4 Demonstration

I plan to demonstrate my idea by implementing my logging library with the identified use case fleshed out. I will then fork an existing open-source code base, refactoring a portion of it to instead use my developed library and then comparing metrics such as lines of code and usability of logs.

5 Measure of Success

My measure of success will deal with identified metrics such as lines of code or usability of logs and the improvement of these metrics from control code versus code that uses my library. Ideally, metrics similar to lines of code would decrease but metrics like usability of logs or trace-ability of errors increase as a result of using my logging library.