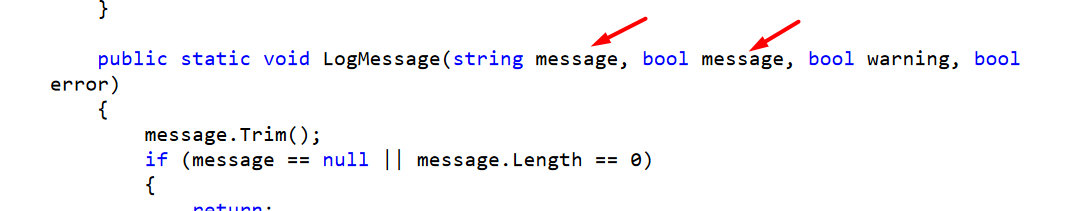
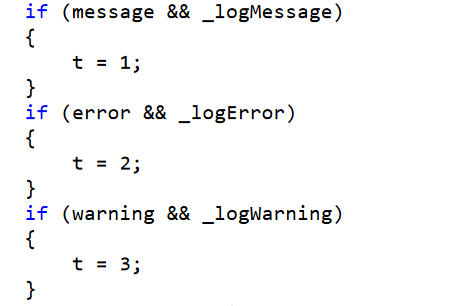
## Name: Eduardo del Castillo Huaccha

JobLogger Code Review Feedback

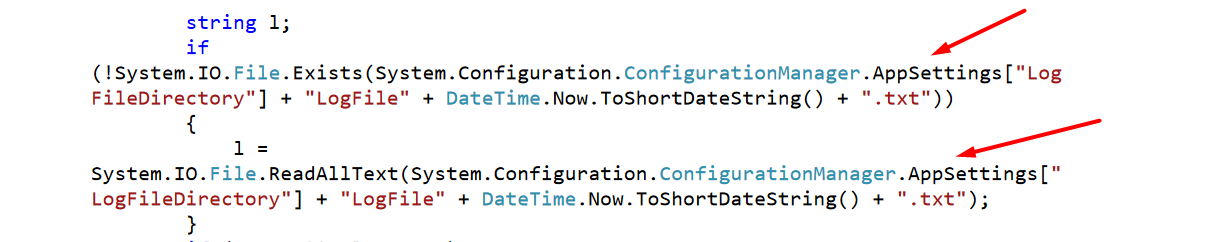
1. The method “JobLogger” does it all, it affects the scalability and maintenance. I suggest to separate this method based on the log level (error, warning, info) and the action (save in file, database or show in console)
2. This only method also has the same name for different arguments, so this will give an error for sure.



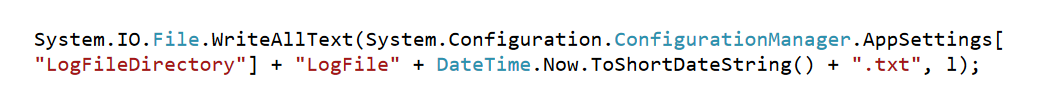
1. It’s highly recommended to use objects, because the logs have attributes in common then it will be easier to set the values on the objects.
2. Another suggestion is to separate the database connection to a different class that handles all the database connections.
3. Duplicated code for if validations.



1. The file path for logging is hardcoded everywhere, this should be taken out, so we can manage this without making an impact in all the code.



1. Also, while file writing the actual code saves all the text again which make it have a poor performance



1. Because of the fixes mentioned so far this will cause us to have small utility methods (to concatenate strings or set objects), that’s why I suggest to have a class that handles these methods.
2. Based on logging best practices, I suggest to add more logging levels, the complete pack Will be: Fatal, Error, Warning, Info, Debug and Trace. This way we will be able to log also the trail of our steps using the system.
3. Also, I suggest to use class Interfaces, then we could just make our logging system more scalable.
4. Finally, and as requested in the documentation, Unit tests are done. This one is an important part to truly trust our code and be confident that it will actually work.