How to properly handle error logs?

Asked 15 years, 9 months ago Modified 15 years, 9 months ago

Viewed 2k times

2

I tried to do several searches before posting this question. If this is a duplicate, please let me know and I will delete it.

My question revolves around the proper way to handle errors produced through our web application. We currently log everything through log4j. If an error happens, it just says "An error has occurred. The IT Department has been notified and will work to correct this as soon as possible" right on the screen. This tells the user nothing... but it also does not tell the developer anything either when we try to reproduce the error. We have to go to the error log folder and try finding this error. Let me also mention that the folder is full of logs from the past week. Each time there is an error, one log file is created for that user and email is sent to the IT staff assigned to work on errors. This email does not mention the log file name but it is a copy of the same error text written that is in the log file.

So if Alicia has a problem at 7:15 with something, but there are 10 other errors that happen that same minute, I have to go through each log file trying to find hers.

What I was proposing to my fellow co-workers is adding an Error Log table into the database. This would write a record to the table for each error, record who it is for, the error, what page it happened on, etc. The bonus of this would be that we can return the primary key value from the table (error_log_id) and show that on the page with a message like "Error Reference Id (1337) has been logged and the proper IT staff has been notified. Please keep this reference id handy for future use". When we get the email, it would tell us the id of the error for quick reference. Or if the user is persistent, they can contact us with the id and we can find the error rather quickly.

How do you setup your error logging? By the way, our system uses Java Servlets that connect to a SQL Server database.

java servlets error-handling

Share

Improve this question

Follow

asked Mar 6, 2009 at 13:24

Ascalonian

15.1k ● 18 ● 72 ● 107

6 Answers

Sorted by: Highest score (default)

I answered a similar question here, but I will adapt that answer to your question.

7




We use **requestID** for this purpose - assign a request ID to each incoming (HTTP) request, at the very beginning of processing (in filter) and then log that on every log line, so you can easily grep those logs later by that ID and find all relevant lines.

If you think it is very tedious to add that ID to every log statement, then you are not alone - java logging frameworks have made it transparent with the use of Mapped Diagnostic Context (MDC) (at least log4j and logback have this).

RequestID can also work as a handy reference number, to spit out, in case of errors (as you already suggested). However, as others have commented, it is not wise to load those details to database - better use file-system. Or, the simplest approach is to just use the requestID - then you do not need to do anything special at the moment error occurs. It just helps you to locate the correct logfile and search inside that file.

How would one requestID look like?

We use the following pattern:

> <instanceName>:<currentTimeInMillis>.
> <counter>

In consists of the following variables:

- *instanceName* uniquely identifies particular JVM in particular deployment environment / .
- *currentTimeInMillis* is quite self-explanatory. We chose to represent it in human-readable format "yyyyMMddHHmmssSSS", so it is easy to read request start time from it (beware: SimpleDateFormat is not thread-safe, so you need to either synchronize it or create a new one on each request).
- *counter* is request counter in that particular millisecond - in the rare case you might need to generate more than one request ID in one millisecond

As you can see, the ID format has been set up in such a way that *currentTimeInMillis.counter* combination is guaranteed to be unique in particular JVM and the whole ID is guaranteed to be globally unique (well, not in the true sense of "global", but it is global enough for our purposes), without the need to involve database or some other central node. Also, the use of *instanceName* variable gives you the possibility to limit the number of log files you later need to look in to find that request.

Then, the final question: *"that is fine and dandy in single-JVM solution, but how do you scale that to several JVMs, communicating over some network protocol?"*

As we use Spring Remoting for our remoting purposes, we have implemented custom RemoteInvocationFactory (that takes request ID from context and saves it to RemoteInvocation attributes) and

RemoteInvocationExecutor (that takes request ID from attributes and adds it to diagnostic context in the other JVM).

Not sure how you would implement it with plain-RMI or other remoting methods.

Share Improve this answer

Follow

edited May 23, 2017 at 12:30

Community Bot

1 ● 1

answered Mar 6, 2009 at 16:29

Neeme Praks

9,130 ● 5 ● 48 ● 50

1

If multiple servers are running and each server leaves log messages on itself, it is really difficult to trace them. So,somebody or a tool should gather and sort them in time order. It is a good way to have a central point where all messages are sent.

Share Improve this answer

Follow

answered Mar 6, 2009 at 14:27

grayger

931 ● 8 ● 20

1

A possible solution, have your error page include a 'send email to whatever' link. When the user clicks this email the body of the e-mail might start with a few blank lines followed by something like:

> ----Please do not modify the information below this line.---
>
> Error details

Any users complaining via this link will automatically send you the info you need and if you are reproducing the error you have quick access to the error message. You might even have a form for sending the e-mail so that the user never sees this (which may be important to some) but then you are relying on your system being at least able to send an e-mail.

Actually I find it useful to print the error details in an HTML comment on error pages like this so that I can always get at them myself.

I do agree with david above that I do not like storing this kind of information in a DB.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

answered Mar 6, 2009 at 15:03

Kris

14.5k ● 7 ● 57 ● 65

For the strategies of logging you can see the discussion Logging best practices.

1

Share Improve this answer

Follow

edited May 23, 2017 at 12:30

Community Bot

1 • 1

answered Mar 6, 2009 at 15:43

alepuzio

1,418 • 2 • 30 • 39

1

I have used an approach like the one you're suggesting (log to a db) in the past and it has been **very** helpful.

Not only you cat get the error via SQL but you can also generate reports of what's the most recurring errors and attend them first.

On the design we did, equals stacktraces belong to the same records (since they were originated exactly in the same place)

We had an small app that pooled that db and we knew then a new exception was generated instead of getting an e-mail that summed with the rest of the previous weeks were ignored altogether.

Of course this database design was very specific for the application we had and additional identifications were possible, we had software version, build, some times input parameters , etc. etc.

With time, the system administrators get to know what to do with each kind of exception and proceed accordingly.

But! Your application may not be that big anyway.
Probably you can have the same just parsing the log files.

Share Improve this answer

Follow

answered Mar 6, 2009 at 16:44

OscarRyz

199k ● 119 ● 396 ● 573

1

I'd oppose the idea of storing error logs in a database. The logging system should be as simple as possible and not involve components that are not 100% necesary to write a log record.

Things can get pretty complex when logging into a DB - e.g. you can having troubles logging any database-related errors (how to log errors that occured because DB not responding, e.g. because of a heavy load or a infrastructure error); other issue I'd see is a potential need to have separate transactions for logging, etc.

On the other hand, having a reference ID for an error is not a bad idea, but again, this it also means to increase complexity of logging system (e.g. how would you propagate the ref. ID through all layers of your application when a error occurs?)

In projects I'm involved to, the general guideline is to log errors as verbosely as possible, and to include as much context information as possible (to write the logs, we use a 'conventional' approach usually - log4j or simillar). Usually, this works well even for heavy loaded systems.

Share Improve this answer Follow

edited Mar 6, 2009 at 19:00

answered Mar 6, 2009 at 13:50

david a.

5,291 ● 24 ● 25