Building Better Log Entries to Enable Faster Analysis

Erik Dahl
PRINCIPAL ARCHITECT

@dahlsailrunner knowyourtoolset.com

Overview

Review Existing Log Entries

- What is missing?
- Any un-necessary information?
- Any duplicated information?

Using Scopes to add "context"

Exception handling revisited

- Detect higher levels

Protect sensitive information

Anatomy of a Log Entry

Field	Description	Sample
Timestamp	Time of the log entry	...
Level	Level of the entry	Information
MessageTemplate	Entry with replaceable placeholders	{UserId} did {ActivityName}
Message	Entry with replacements made	123 did Book Submission
SourceContext	Category for the log entry	BookClub.API.Controllers.BookController
ActionId	Identifier for the Action (spans requests)	4348b30c-93de-48e8-8eec-2b680c0311de
ActionName	Fully-qualified namespace / class / method for action	BookClub.API.Controllers.BookController.GetBooks (BookClub.API)
RequestId	Unique id for the request	80000017-0002-f700-b63f-84710c7967bb
RequestPath	Local path for the request	/api/Book
CorrelationId	Something like session id to track user activity within a session	
<replacement values>	Replacement values for MessageTemplate	
Exception	ToString() representation	...

What's Missing?

Field
Timestamp
Level
MessageTemplate
Message
SourceContext
ActionId
ActionName
RequestId
RequestPath
CorrelationId
<replacement values>
Exception

- Machine name
- Environment
- User Id
- Role(s)
- Entry Assembly
- Version
- Customer Id
- HttpContext

- Include what's important to YOU

Key Points

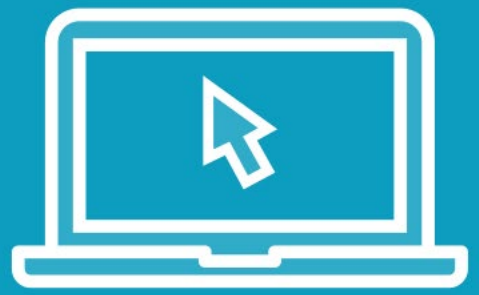
DO consider what will truly assist in troubleshooting / analysis

DON'T log what you don't need to log – it can be expensive!

NOTE: Excluding the standard info is harder

DON'T add information that repeats already-included info

Demo



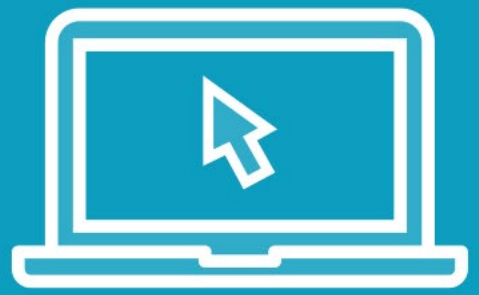
Add following information to entries:

- UserId
- Scopes (OAuth2)
- Machine
- Entry Assembly

Use Scope to add the info

- Go back to performance filters
- Add scope info
- Verify results

Demo



Detect higher exception levels

- Is application completely down or did a user experience an error?

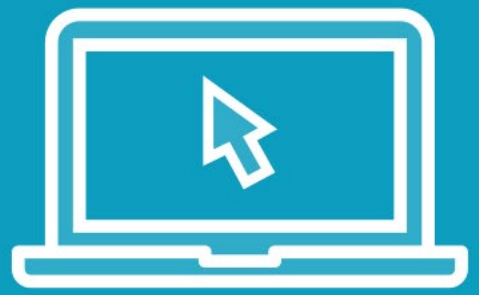
Can be used for alerting

Requires custom code or policies

Our example:

- Database down / unreachable versus missing proc

Demo



Don’t log sensitive information!

- Know what constitutes “sensitive”

Enable troubleshooting, but protect information

Do the protection at point-of-logging

- “Formatters” could help you centralize this
- Probably based on “field name”

Our example:

- Include masked email address

Summary

Looked closely at log entry content

- Considered missing content
- Discussed content to omit

Added missing information using Scopes

Revisited exception handling

- Logged Fatal errors on certain condition

Discussed sensitive information

Next: Enabling consumption