Implementing Cross-cutting Concerns for ASP.NET Core Microservices

IMPLEMENTING LOGGING

Steve Gordon
MICROSOFT DEVELOPER TECHNOLOGIES MVP

@stevejgordon www.stevejgordon.co.uk

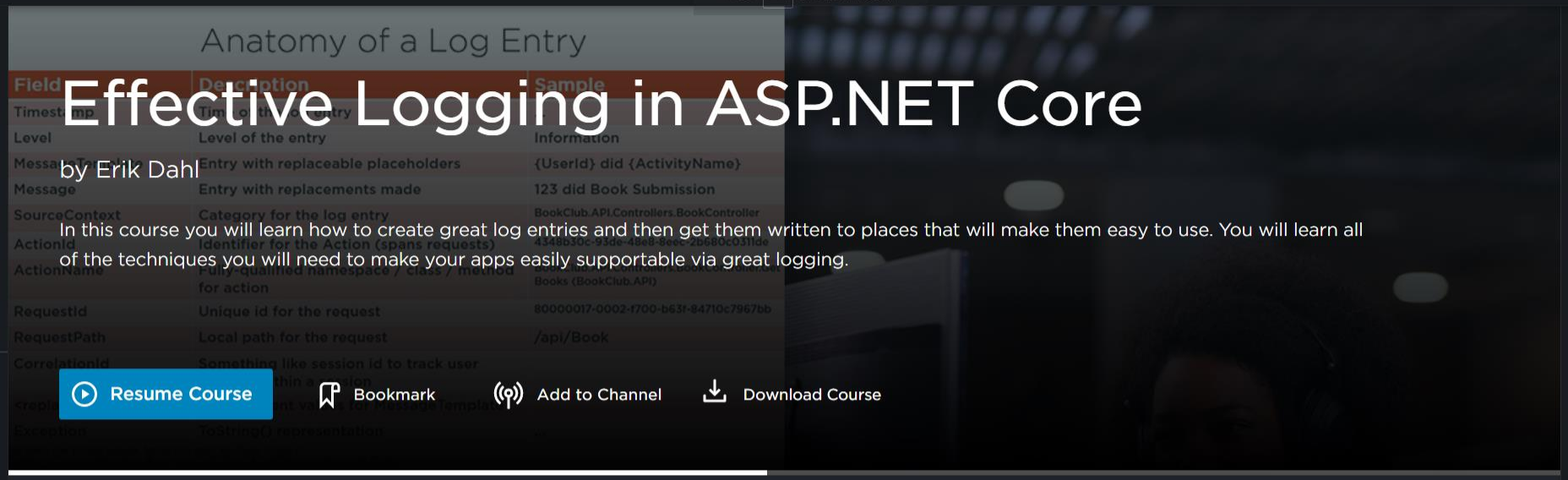
Overview

- Learn why logging is often crucial in distributed systems
- Discuss challenges of logging when using microservice architectures
- Consider what information we should log
- Learn about ASP.NET Core logging
- Implement initial logging
- Support code reuse with a shared library

Later in This Course

Implementing centralized logging in .NET Core microservices

Implementing health checks in ASP.NET Core microservices



Effective Logging in ASP.NET Core

by Erik Dahl

In this course you will learn how to create great log entries and then get them written to places that will make them easy to use. You will learn all of the techniques you will need to make your apps easily supportable via great logging.

Resume Course | Bookmark | Add to Channel | Download Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Related Courses

This course is part of: ASP.NET Core Path

Expand All

Course Overview	1m 18s
Logging in ASP.NET Core Quickstart	30m 12s
Controlling What Messages are Logged in ASP.NET Core Applications	31m 10s
Automating Logging of Standard Events in ASP.NET Core	36m 45s
Building Better Log Entries to Enable Faster Analysis	21m 26s
Enabling Consumption	25m 36s

Course author

Erik Dahl

Erik Dahl has been developing software and architecture for 20+ years, mostly doing in-house development for his employers. His recent work has included a multi-tenant B2B implementation and...

Course info

Level	Intermediate
Rating	★★★★☆ (70)
My rating	★★★★★
Duration	2h 26m
Updated	29 Jul 2020

Share course

GLOBOTICKET
A Globomantics Company

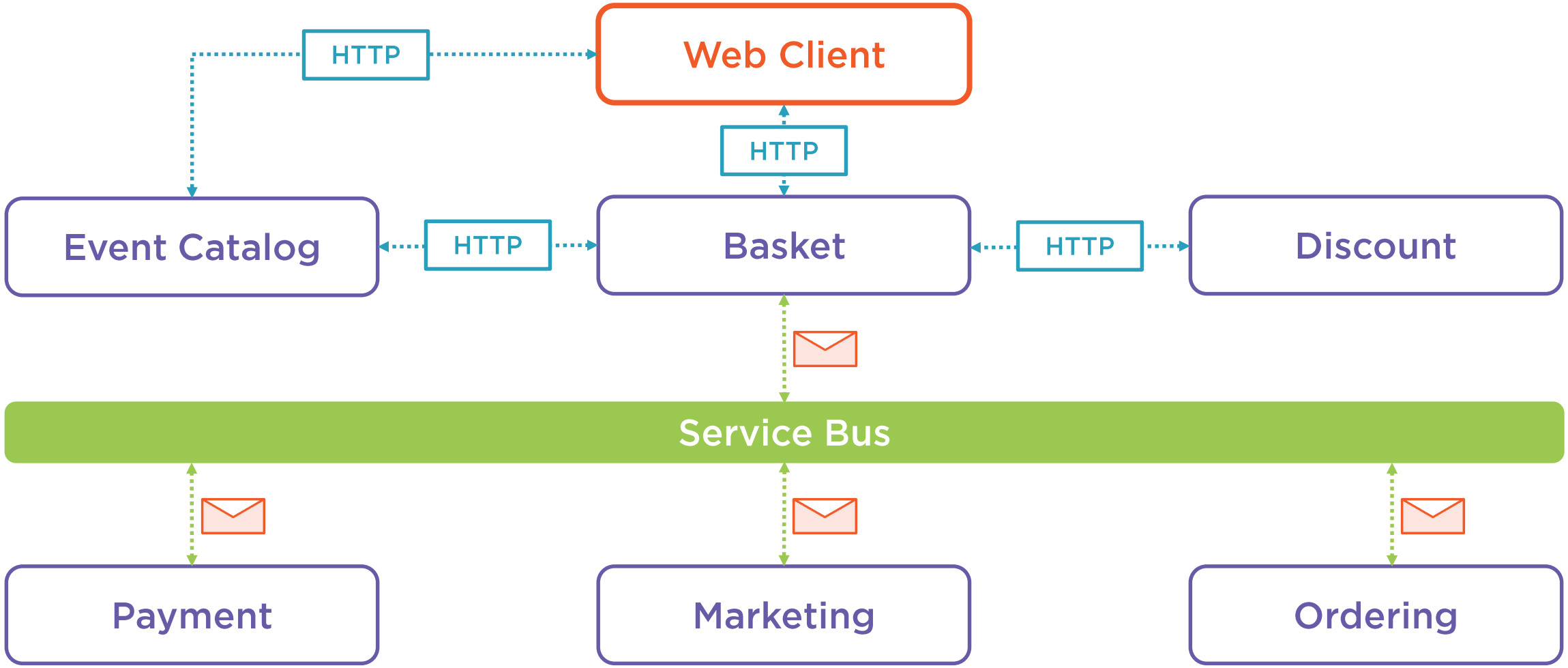
0 TICKETS

All

FILTER

	DATE	NAME	ARTIST	PRICE	
	09/05/2021	To the Moon and Back	Nick Sailor	$135	DETAILS
	09/01/2021	Clash of the DJs	DJ 'The Mike'	$85	DETAILS
	09/07/2021	Techorama 2021	Many	$400	DETAILS
	09/01/2021	Spanish guitar hits with Manuel	Manuel Santinonisi	$25	DETAILS
	09/03/2021	John Egbert Live	John Egbert	$65	DETAILS
	09/06/2021	The State of Affairs: Michael Live!	Michael Johnson	$85	DETAILS

Overall Application Architecture



ASP.NET Core Microservices: Getting Started

by Roland Guijt

Using microservices in the architecture of the application is both powerful and complex. This course will teach you why that is and how to get started creating microservices using ASP.NET Core.

Start Course | Bookmark | Add to Channel | Download Course

Table of contents | Description | Transcript | Exercise files | Discussion | Related Courses

Expand All

Course Overview	1m 17s
How to Create a Microservice	28m 3s
Connecting Microservices Synchronously and Asynchronously	26m 15s
Microservices Considerations and Design	25m 37s

Course author

Roland Guijt

Roland is a Microsoft MVP enjoying a constant curiosity around new techniques in software development. His focus is on all things .Net and browser technologies. As a long-time trainer, he led many...

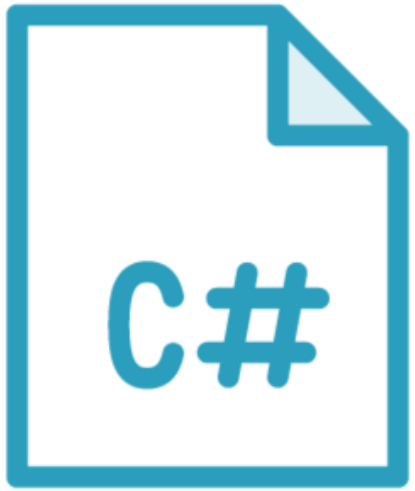
Course info

Level	Beginner
Rating	★★★★★
My rating	★★★★★
Duration	1h 21m
Released	24 Sep 2020

Share course

Course Prerequisites

Some knowledge of ASP.NET Core



Experience with C#

Before We Begin

Follow along: Download the exercise files

The solution requires the .NET Core 3.1 SDK

I'm using Visual Studio 2019 (16.7.x)

Let’s Get Started

The Importance of Logging

Logging

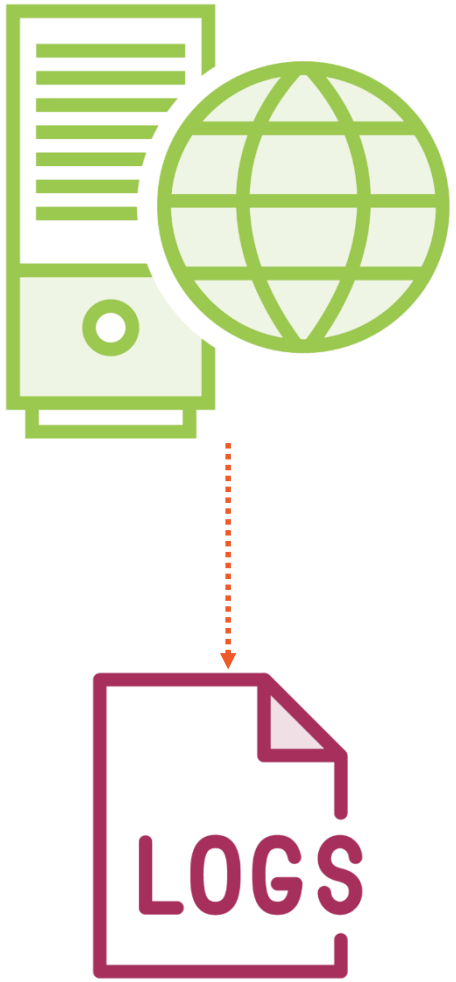
Record operational events at runtime

Used to:

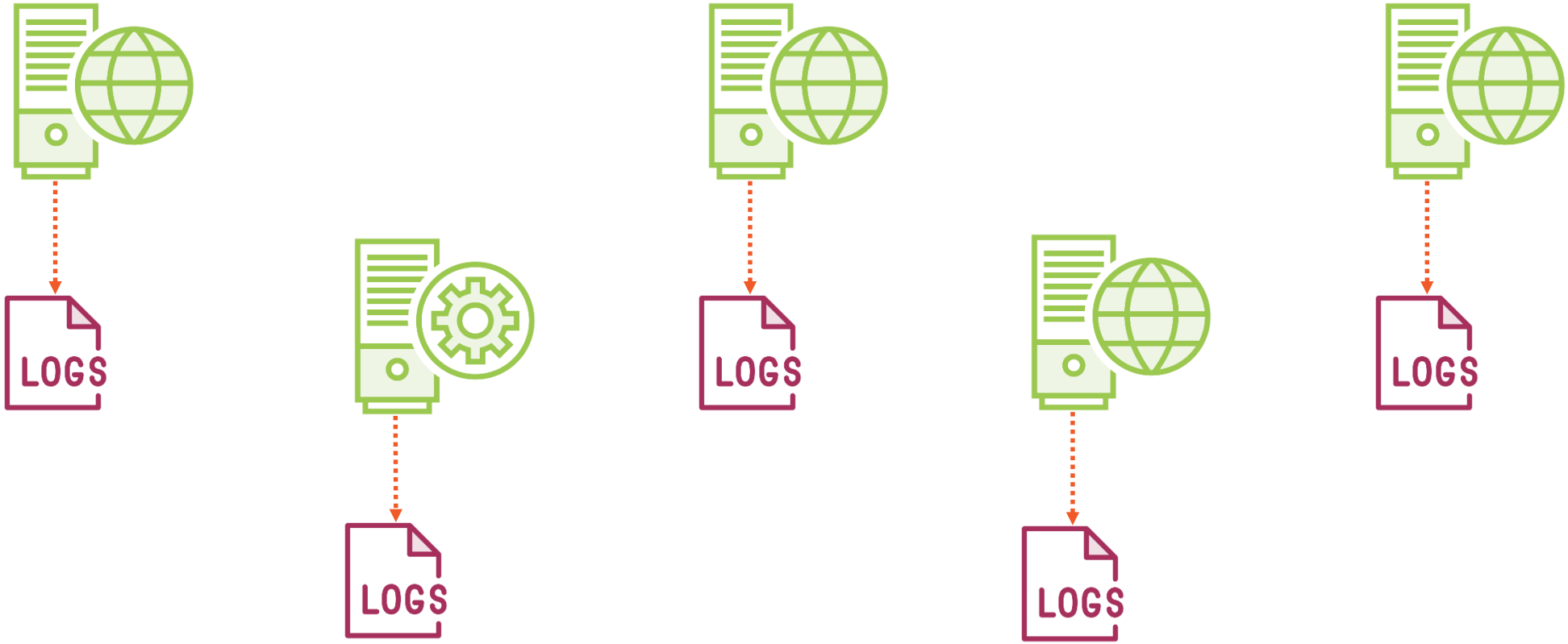
- Understand behaviour of a service
- Identify errors for investigation
- Diagnose bugs and failures

It is important to log information you may later depend on

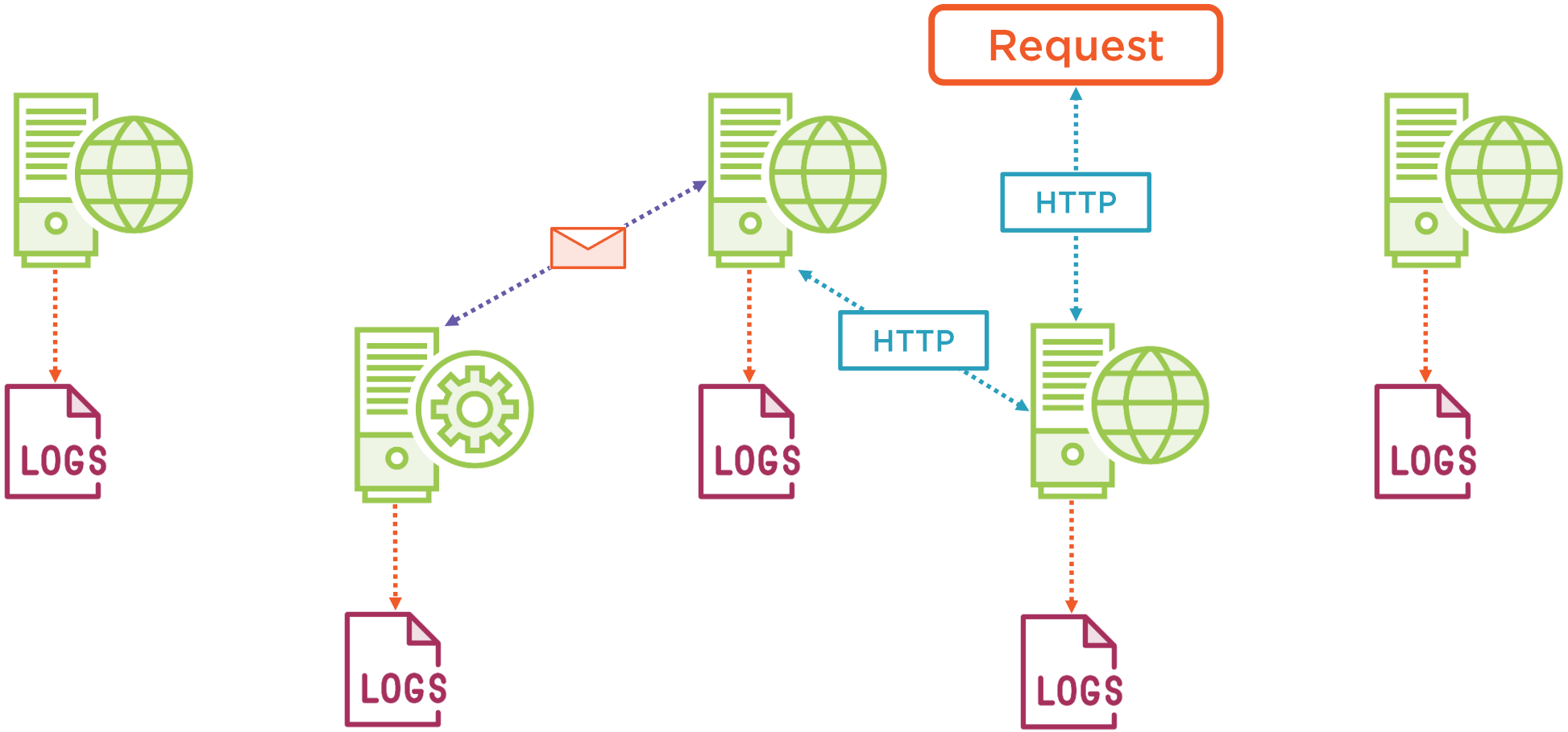
Monolithic Application



Microservices Application

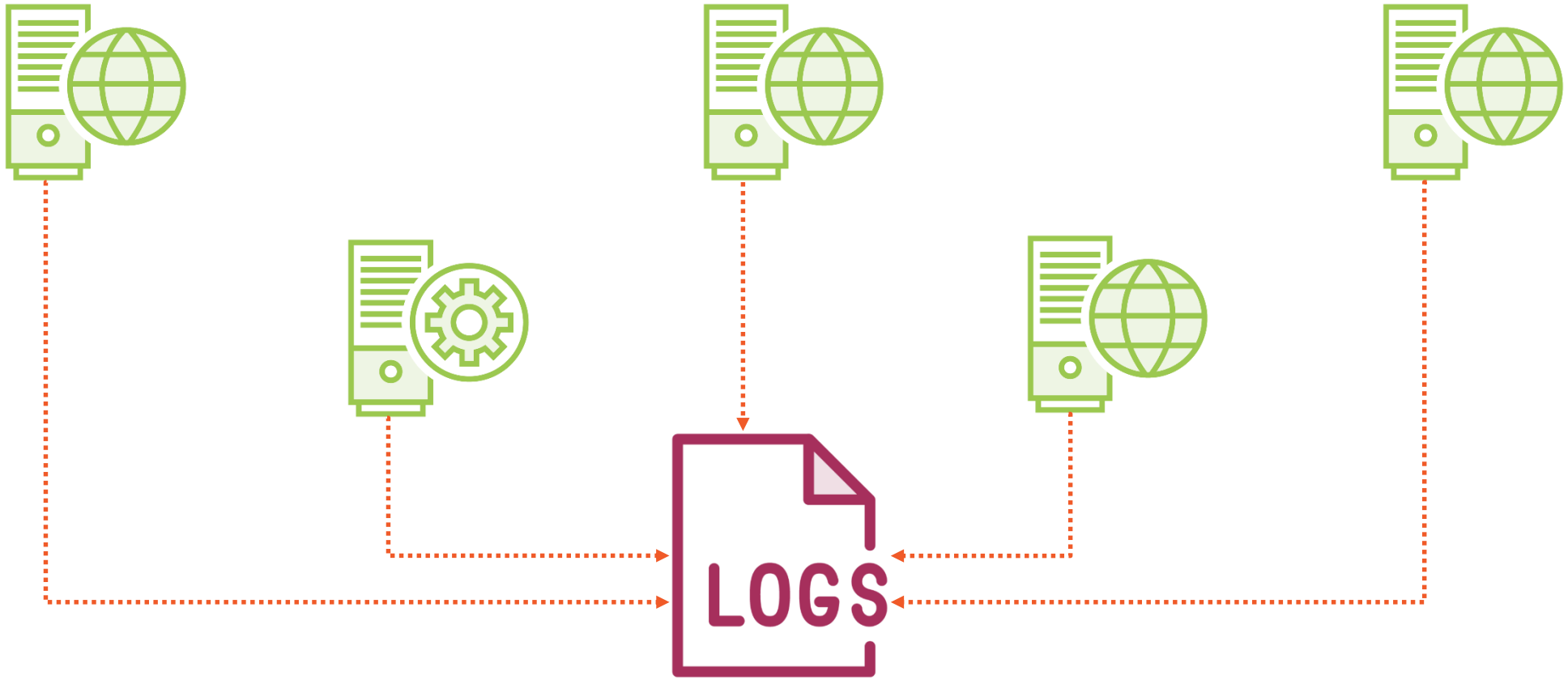


Microservices Application



?

Microservices Application



Choosing What to Log

Logging Requirements

What information will be needed to diagnose a bug or runtime error?

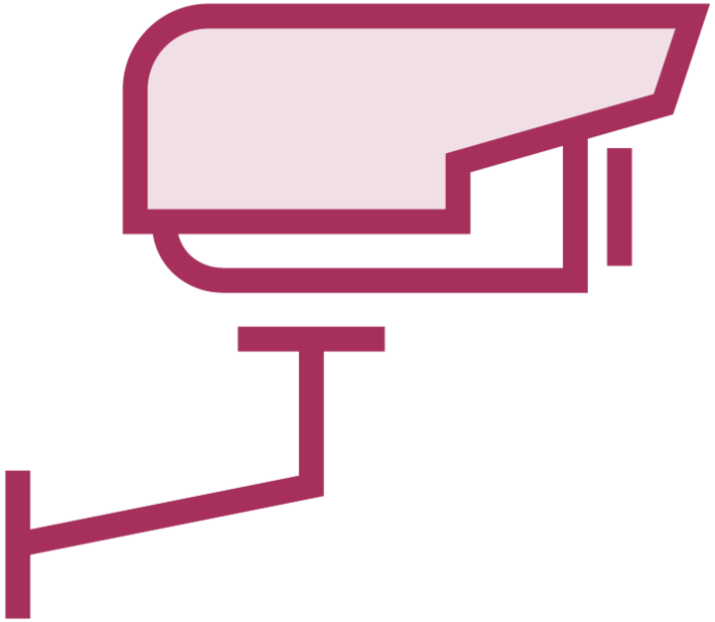
Balance between logging too much or too little

- Log enough to be useful
- Avoid introducing redundant noise

Messages should contain enough detail and data to support proper analysis

- For example, record request and/or resource IDs

Security



Avoid logging sensitive information

- Passwords
- Credit card details
- Personally identifiable information (PII)

Store IDs of resources

- Look up data when reviewing logs
- No need to manage data deletion
- Reduce storage costs of duplicated data

Microsoft Logging Abstractions

Microsoft Logging Abstractions

Introduced with ASP.NET Core 1.0

Code against a simple interface

Plug-in third-party logging frameworks

ILogger<T>

```
namespace Microsoft.Extensions.Logging
{
    public interface ILogger<out TCategoryName> : ILogger
    {
    }
}
```

ILogger

```
namespace Microsoft.Extensions.Logging
{
    public interface ILogger
    {
        void Log<TState>(LogLevel logLevel, EventId eventId, TState state,
                    Exception exception, Func<TState, Exception, string> formatter);

        bool IsEnabled(LogLevel logLevel);

        IDisposable BeginScope<TState>(TState state);
    }
}
```

ILoggerFactory

```
namespace Microsoft.Extensions.Logging
{
    public interface ILoggerFactory : IDisposable
    {
        ILogger CreateLogger(string categoryName);
        void AddProvider(ILoggerProvider provider);
    }
}
```

Demo

Debug the GloboTicket application

Explore default log messages

Log Categories

ILogger instances require a category name

Categories are included in log messages

Support filtering and grouping

ILogger<T> Log Category

HomeController.cs

```
public class HomeController : Controller
{
    public HomeController(ILogger<HomeController> logger) { ... }
}
```

ILogger<T> Log Category

HomeController.cs

```
public class HomeController : Controller
{
    public HomeController(Ilogger<HomeController> logger) { ... }
}
```

```
error: MyApplication.Controllers.HomeController
       Something failed!
```

Log Levels

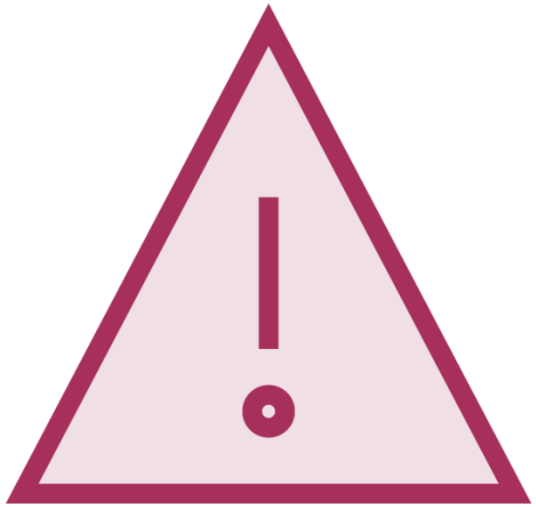
Log Levels

Tag messages with metadata about the importance of the event

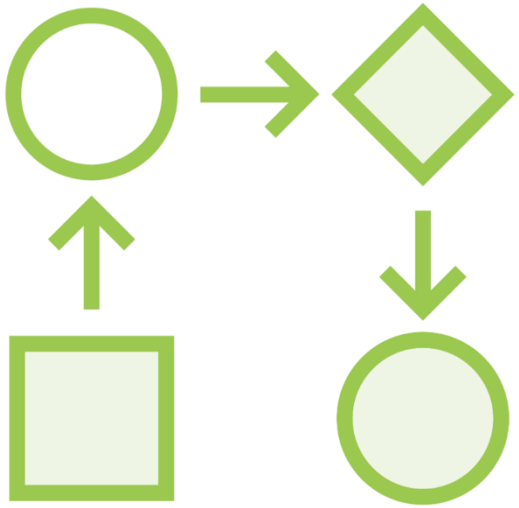
Each message includes a log level

Log messages can be filtered based on their log level

Filtering by Log Level



Always log errors and exceptions



Sometimes log conditional application flow

Microsoft Log Levels

Level	Usage
Trace	Log detailed messages during development
Debug	Log verbose messages (occasionally in production)
Information	Log general flow of requests/operations
Warning	Log non-critical but abnormal events
Error	Log exceptions which cannot/are not gracefully handled
Critical	Log major failures which require immediate attention

Demo

Filter console log messages

- Use configuration to filter logs from categories by log level

Using Configuration and Options in .NET Core and ASP.NET Core Apps

by Steve Gordon

This course will teach you everything you need to know about using configuration and options in ASP.NET Core. The skills you will learn will help you to build complex ASP.NET Core applications which can be configured from multiple sources.

Resume Course | Bookmarked | Add to Channel | Download Course

Table of contents | Description | Transcript | Exercise files | Discussion | Learning Check | Related Courses

This course is part of: ASP.NET Core Path

Expand All

Course Overview	2m 2s	
Getting Started with Configuration Concepts	34m 48s	
Applying the Options Pattern	54m 41s	
Working with Configuration Providers	38m 43s	

Course author

Steve Gordon

Steve Gordon is a Microsoft MVP, senior developer and community lead based in Brighton, UK. He works for Madgex developing and supporting their data products portfolio, built using .NET Core...

Course info

Level	Intermediate
Rating	★★★★☆ (102)
My rating	★★★★★
Duration	2h 10m
Released	23 Sep 2019

Share course

Demo

Record application log messages from the GloboTicket web application

Update configuration to enable debug logging during development

Default Logging

Default messages are verbose and low-level

All messages use information log level

- Challenging to filter

Demo

Logging application exceptions

- Add a try/catch block
- Include the exception details in logs

Demo

Focus on code reuse

Add a shared library project

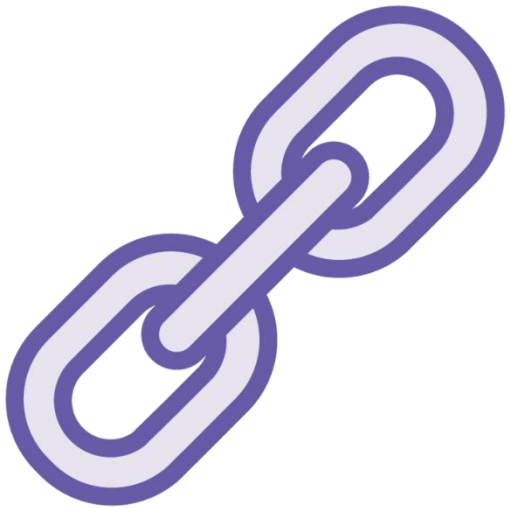
Pros and Cons of Shared Libraries

Shared Libraries

Pros	Cons
Reduce development time	May reintroduce coupling
Code is written and tested once	Increased maintenance complexity
Bugs can be fixed in a single place	Must consider versioning and updates
	Microservice releases may be locked together

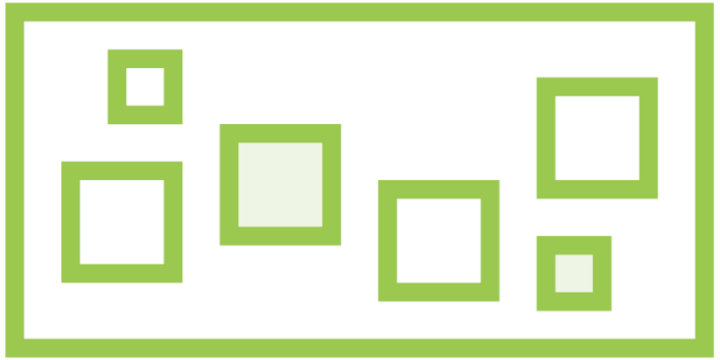
Alternatives

Copy/paste code



Shared project

Microservice Considerations



Avoid shared libraries for business logic

- Code duplication may be a better choice

Ensure services have a single responsibility

- Consider a shared microservice, responsible for common business logic

Consider shared libraries for infrastructure and utility code reuse

Summary

- Learned about logging challenges in microservices
- Introduced the Microsoft logging abstractions
- Learned about log levels and categories
- Filtered log output using configuration
- Implemented application logging
- Shared common code in a library

Up Next:

Implementing Centralized Logging for Microservices