Question

Case Study: Order & Payment Microservices with Zipkin Tracing

Business Scenario

An e-commerce company wants to track how requests flow through its microservices to improve
debugging and performance monitoring.
Currently, when a customer places an order, the request goes through multiple services, and it's hard
to identify delays or errors without detailed logging.

They decide to implement Spring Cloud Sleuth and Zipkin for distributed tracing.

System Architecture

Microservices:

1. Order Service
◦ Receives customer orders.
◦ Calls Payment Service to process payment.
◦ Responds back to the customer with order confirmation.

2. Payment Service
◦ Receives payment requests from Order Service.
◦ Simulates payment processing.
◦ Sends confirmation back to Order Service.

Monitoring Tool:

• Zipkin Server
◦ Collects and displays tracing data from all services.
◦ Shows the timeline, latency, and flow of each request.

Workflow

1. Customer sends a POST /orders request to Order Service.
2. Order Service:
◦ Generates a trace ID (via Spring Cloud Sleuth).
◦ Logs the start of the process.
◦ Calls the Payment Service using REST.

3. Payment Service:
◦ Receives the request with the same trace ID.
◦ Processes the payment.
◦ Logs the completion.

4. The trace data is sent to Zipkin.

5. The developer opens the Zipkin UI (http://localhost:9411) to:
◦ Search by trace ID.
◦ View service-to-service request flow.
◦ Analyze request duration and bottlenecks.

Sample Trace in Zipkin UI

Zipkin visually shows that the Payment Service took 80% of the total request time, so the dev
team can investigate payment processing delays.

Real-World Analogy

Think of this like a courier tracking system:
• You send a package (request) from your house (Order Service).
• It goes through a distribution hub (Payment Service).
• Every checkpoint logs the same tracking number (trace ID).
• At any time, you can check the courier website (Zipkin UI) to see where delays occurred.

Service Operation Duration

Order Service Create Order 150 ms
Payment Service Process Payment 120 ms

http://localhost:9411/

Code

```
// Order Entity
package com.example.zipkin.entity;

public class Order {
   private String orderId;
   private String customerName;
```

```
    private double amount;

    // Getters and Setters
    public String getOrderId() { return orderId; }
    public void setOrderId(String orderId) { this.orderId = orderId; }
    public String getCustomerName() { return customerName; }
    public void setCustomerName(String customerName) { this.customerName =
customerName; }
    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }

    @Override
    public String toString() {
        return "Order{" +
                "orderId='" + orderId + '\'' +
                ", customerName='" + customerName + '\'' +
                ", amount=" + amount +
                '}';
    }
}

// Order Controller (Order Service)
package com.example.zipkin.controller;

import com.example.zipkin.entity.Order;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

@RestController
@RequestMapping("/orders")
public class OrderController {

    @Autowired
    private RestTemplate restTemplate;

    @PostMapping
    public String placeOrder(@RequestBody Order order) {
        System.out.println("Order Service - Received Order: " + order);
        String response = restTemplate.postForObject("http://localhost:8082/payments",
order, String.class);
        return "Order placed successfully! Payment Response: " + response;
    }
```

```
}

// Payment Controller (Payment Service)
package com.example.zipkin.controller;

import com.example.zipkin.entity.Order;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/payments")
public class PaymentController {

  @PostMapping
  public String processPayment(@RequestBody Order order) {
    System.out.println("Payment Service - Processing Payment for Order: " + order);
    return "Payment of " + order.getAmount() + " for Order " + order.getOrderId() + " is
successful!";
  }
}

// Configuration for RestTemplate Bean (Order Service)
package com.example.zipkin.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class AppConfig {

  @Bean
  public RestTemplate restTemplate() {
    return new RestTemplate();
  }
}

// application.properties for both services
spring.application.name=order-service
server.port=8081

# Zipkin Configuration
spring.zipkin.base-url=http://localhost:9411/
spring.sleuth.sampler.probability=1.0
```

```
spring.application.name=payment-service
server.port=8082

# Zipkin Configuration
spring.zipkin.base-url=http://localhost:9411/
spring.sleuth.sampler.probability=1.0
```