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

//payment.java

```
package com.example.zipkinpaymentservice.model;

public class Payment {
    private Long orderId;
    private double amount;

    public Payment(Long orderId, double amount) {
        this.orderId = orderId;
        this.amount = amount;
    }

    public Long getOrderId() {
        return orderId;
    }
    public double getAmount() {
        return amount;
    }
}
```

//PaymentController

```
package com.example.zipkinpaymentservice.controller;
```

```
import com.example.zipkinpaymentservice.model.Payment;
import com.example.zipkinpaymentservice.service.PaymentService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class PaymentController {

    private final PaymentService paymentService;

    public PaymentController(PaymentService paymentService) {
        this.paymentService = paymentService;
    }

    @PostMapping("/payments")
    public String processPayment(@RequestBody Payment payment) {
        return paymentService.processPayment(payment);
    }

    @GetMapping("/payments")
    public List<Payment> getPayments() {
        return paymentService.getPayments();
    }
}
```

//PaymentService

```
package com.example.zipkinpaymentservice.service;
```

```
import com.example.zipkinpaymentservice.model.Payment;
import io.micrometer.tracing.Tracer;
import io.micrometer.tracing.Span;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class PaymentService {

    private final Tracer tracer;
    private final List<Payment> payments = new ArrayList<>(); // in-memory storage

    public PaymentService(Tracer tracer) {
        this.tracer = tracer;
    }

    public String processPayment(Payment payment) {
        payments.add(payment); // store payment

        Span newSpan = tracer.nextSpan().name("process-payment").start();
        try (Tracer.SpanInScope ws = tracer.withSpan(newSpan)) {
            return "Payment processed for Order ID: " + payment.getOrderId();
        } finally {
            newSpan.end();
        }
    }

    public List<Payment> getPayments() {
        return payments;
```

```
    }
}
```

//application.properties

```
spring.application.name=zipkin-payment-service
server.port=8082
management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans
management.tracing.sampling.probability=1.0
```

//AppConfig

```
package com.example.zipkinordersservice.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();
    }
}
```

//OrderController

```
package com.example.zipkinordersservice.controller;


import java.util.List;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.zipkinordersservice.model.Order;
import com.example.zipkinordersservice.service.OrderService;

@RestController
public class OrderController {

    private final OrderService orderService;

    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }

    @GetMapping("/orders")
    public List<Order> getOrders() {
        return orderService.getOrders();
    }



    @PostMapping("/orders")
    public String placeOrder(@RequestBody Order order) {
        return orderService.placeOrder(order);
    }
}

//Order.java
package com.example.zipkinordersservice.model;
```

```
public class Order {
    private Long id;
    private String productName;
    private double price;

    public Order(Long id, String productName, double price) {
        this.id = id;
        this.productName = productName;
        this.price = price;
    }

    public Long getId() {
        return id;
    }
    public String getProductName() {
        return productName;
    }
    public double getPrice() {
        return price;
    }
}
```

//OrderService

```
package com.example.zipkinordersservice.service;

import io.micrometer.tracing.Tracer;
import io.micrometer.tracing.Span;

import java.util.ArrayList;
```

```
import java.util.List;

import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.example.zipkinordersservice.model.Order;

@Service
public class OrderService {

   private final RestTemplate restTemplate;
   private final Tracer tracer;
   private final List<Order> orders = new ArrayList<>();

   public OrderService(RestTemplate restTemplate, Tracer tracer) {
      this.restTemplate = restTemplate;
      this.tracer = tracer;
   }

   public String placeOrder(Order order) {
      orders.add(order); // save order in memory

      Span newSpan = tracer.nextSpan().name("place-order").start();
      try (Tracer.SpanInScope ws = tracer.withSpan(newSpan)) {
         String paymentServiceUrl = "http://localhost:8082/payments";
         String paymentResponse = restTemplate.postForObject(paymentServiceUrl, order,
String.class);
         return "Order placed for: " + order.getProductName() + ". " + paymentResponse;
      } finally {
         newSpan.end();
      }
```

```
    }

    public List<Order> getOrders() {
        return orders;
    }
}
```

//Application.properties

```
spring.application.name=zipkin-orders-service
server.port=8081
management.zipkin.tracing.endpoint=http://localhost:9411/api/v2/spans
management.tracing.sampling.probability=1.0
```

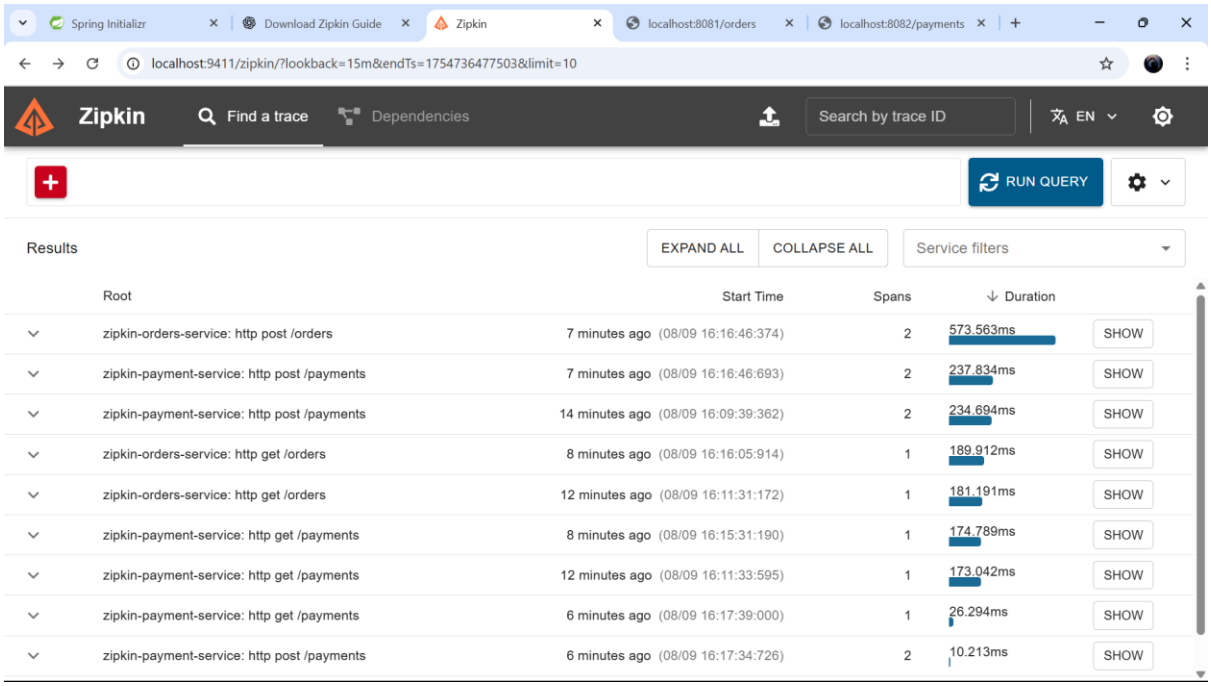
http://localhost:8082/payments

```
{
  "orderId": 1,
  "amount": 85000
}
```

http://localhost:8081/orders

```
  "id": 1,
  "productName": "Laptop",
```

"price": 85000


Spring Initializr
Download Zipkin Guide
Zipkin
localhost:8081/orders
localhost:8082/payments
localhost:9411/zipkin/?lookback=15m&endTs=1754736477503&limit=10
Zipkin
Find a trace
Dependencies
Search by trace ID
EN
RUN QUERY
Results
EXPAND ALL
COLLAPSE ALL
Service filters
Root
Start Time
Spans
Duration
zipkin-orders-service: http post /orders 7 minutes ago (08/09 16:16:46:374) 2 573.563ms SHOW
zipkin-payment-service: http post /payments 7 minutes ago (08/09 16:16:46:693) 2 237.834ms SHOW
zipkin-payment-service: http post /payments 14 minutes ago (08/09 16:09:39:362) 2 234.694ms SHOW
zipkin-orders-service: http get /orders 8 minutes ago (08/09 16:16:05:914) 1 189.912ms SHOW
zipkin-orders-service: http get /orders 12 minutes ago (08/09 16:11:31:172) 1 181.191ms SHOW
zipkin-payment-service: http get /payments 8 minutes ago (08/09 16:15:31:190) 1 174.789ms SHOW
zipkin-payment-service: http get /payments 12 minutes ago (08/09 16:11:33:595) 1 173.042ms SHOW
zipkin-payment-service: http get /payments 6 minutes ago (08/09 16:17:39:000) 1 26.294ms SHOW
zipkin-payment-service: http post /payments 6 minutes ago (08/09 16:17:34:726) 2 10.213ms SHOW

}