Zipkin 服务跟踪

为什么要使用服务跟踪?

我们知道一个分布式系统中，往往会部署很多个微服务，这些微服务之间会有相互调用的关系，整个流程非常复杂，在进行问题排查和性能优化的时候工作量就会比较大，如果我们能够准确地追踪到每个网络请求，了解它的整个运行流程，经过了哪些微服务，是否有延迟，耗费时间等，这样的话我们分析系统性能，排查解决问题就会容易很多，我们使用 Zipkin 组件来实现服务跟踪。

什么是 Zipkin?

Zipkin 是 Twitter 开源的一款分布式实时数据跟踪系统，基于 Googel Dapper 的论文设计而来，其主要功能是聚集来自各个异构系统的实时监控数据。Zipkin 可以分为两部分，一部分是 Zipkin Server，用来作为数据的采集存储、数据分析与展示。Zipkin Client 是 Zipkin 基于不同的语言及框架封装的一系列客户端工具，这些工具完成了追踪数据的生成与上报功能。

Spring Cloud 为服务跟踪提供了解决方案，Spring Cloud 集成了 Zipkin 组件。

代码实现

搭建 Zipkin Server。

- pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>io.zipkin.java</groupId>
        <artifactId>zipkin-server</artifactId>
        <version>2.9.4</version>
    </dependency>

    <dependency>
        <groupId>io.zipkin.java</groupId>
        <artifactId>zipkin-autoconfigure-ui</artifactId>
        <version>2.9.4</version>
    </dependency>
</dependencies>
```

- application.yml，添加 Zipkin 相关配置。

```
server:
  port: 9090
```

- 创建启动类

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import zipkin.server.internal.EnableZipkinServer;

@SpringBootApplication
@EnableZipkinServer
public class ZipkinApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZipkinApplication.class,args);
    }
}
```

搭建 Zipkin Client。

- pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-zipkin</artifactId>
    </dependency>
</dependencies>
```

- application.yml

```
server:
  port: 8090
spring:
  application:
    name: zipkinclient
  sleuth:
    web:
      client:
        enabled: true
    sampler:
      probability: 1.0
  zipkin:
    base-url: http://localhost:9090/
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

`spring.sleuth.web.client.enabled`：设置开启 Sleuth。

`spring.sleuth.sampler.probability`：设置采样比例，默认是 1.0。

`spring.zipkin.base-url`：Zipkin Server 地址。

- 启动类

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ZipkinClientApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZipkinClientApplication.class,args);
    }
}
```

- Handler

```
package com.southwind.controller;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/zipkin")
public class ZipkinHandler {

    @Value("${server.port}")
    private String port;

    @GetMapping("/index")
    public String index(){
        return this.port;
    }
}
```

外卖点餐系统

客户端：用户登录，用户退出，菜品订购，我的订单。

后台管理系统：管理员登录，管理员退出，添加菜品，查询菜品，修改菜品，删除菜品，订单处理，添加用户，查询用户，删除用户。

服务拆分，4个服务提供者。

accout 提供账户服务：用户和管理员的登录，退出。

menu 提供菜品服务：添加菜品，查询菜品，修改菜品，删除菜品。

order 提供订单服务：添加订单，查询订单，删除订单，处理订单。

user 提供用户服务：添加用户，查询用户，删除用户。

1个服务消费者，包括客户端的前端页面和后台接口，后台管理系统的前端页面和后台管理系统，用户 / 管理员直接访问的资源都保存服务消费者中，然后服务消费者调用4个服务提供者对应的接口完成业务逻辑，并且通过 Feign 实现负载均衡。

4个服务提供者和1个服务消费者都需要在注册中心进行注册，同时要注册配置中心，提供远程配置信息读取，服务提供者和服务消费者的配置信息都保存在 Git 仓库，由配置中心负责拉取。

本系统由8个模块组成，分别是注册中心，配置中心，Git 仓库配置信息，服务消费者，4个服务提供者。

数据库

分5张表，分别是：

t_admin：保存管理员数据。

t_menu：保存菜品数据。

t_order：保存订单数据。

t_type：保存菜品分类数据。

t_user：保存用户信息。