原创 使用Zuul聚合微服务

2018-10-27 11:20:06 StarskyBoy 阅读数 317 更多

在很多次场景下，外部请求需要查询Zuul后端的多个微服务。举个例子，一个电影售票手机APP，在购票[illegible]上，既需要查询“电影微服务”获得电影信息，又需要查询“用户微服务”获得当前用户的信息。如果让手机端直接请求各个微服务（即使使用Zuul[illegible]转发），那么网络开销、流量耗费、耗费无法令人满意。那么对于这种场景，可使用Zuul聚合微服务请求——手机APP只需发送一个请求给Zuul，[illegible]Zuul请求用户微服务以及电影微服务，并给手机APP。

使用这种方式，手机端只须发送一次请求即可，简化了客户端侧的开发；不仅如此，由于Zuul、用户微服[illegible]影微服务一般都在同一局域网，因此速效率会非常高。

下面围绕以上这个场景，来编写代码。

在本例中，使用了RxJava结合Zuul来实现微服务请求的聚合。

1.复制项目

复制cloud-register-gateway-zuul微服务修改为cloud-register-gateway-zuul-aggregation微服务。

2.修改启动类

```
@EnableZuulProxy
@SpringBootApplication
public class RegisterGatewayZuulAggregationApplication {

  public static void main(String[] args) {
    SpringApplication.run(RegisterGatewayZuulAggregationApplication.class, args);
  }

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

3.创建实体类

```
@Data
public class User {

   private int id;

   private String userName;

   private String passWord;
}
```

4.创建Java类，名为AggregationService

```
@Service
public class AggregationService {
   @Autowired
   private RestTemplate restTemplate;
```

```
    @HystrixCommand(fallbackMethod = "fallback")
    public Observable<User> getUserById(Long id) {
        // 创建一个被观察者
        return Observable.create(observer -> {
            // 请求用户微服务的/{id}端点
            User user = restTemplate.getForObject("http://cloud-register-user/{id}", User.class, id);
            observer.onNext(user);
            observer.onCompleted();
        });
    }

    @HystrixCommand(fallbackMethod = "fallback")
    public Observable<User> getMovieUserByUserId(Long id) {
        return Observable.create(observer -> {
            // 请求未创建服务的/user/{id}端点，本cloud没有这个微服务你可以自行创建
            User otherUser = restTemplate.getForObject("http://cloud-consumer-user/user/{id}", User id);
            observer.onNext(otherUser);
            observer.onCompleted();
        });
    }

    public User fallback(Long id) {
        User user = new User();
        user.setId(-1);
        return user;
    }
}
```

5.创建Controller

```
@RestController
public class AggregationController {

    public static final Logger LOGGER = LoggerFactory.getLogger(AggregationController.class);

    @Autowired
    private AggregationService aggregationService;

    @GetMapping("/aggregate/{id}")
    public DeferredResult<HashMap<String, User>> aggregate(@PathVariable Long id) {
        Observable<HashMap<String, User>> result = this.aggregateObservable(id);
        return this.toDeferredResult(result);
    }

    public Observable<HashMap<String, User>> aggregateObservable(Long id) {
        // 合并两个或者多个Observables发射出的数据项，根据指定的函数变换它们
        return Observable.zip(
                this.aggregationService.getUserById(id),
                this.aggregationService.getMovieUserByUserId(id),
                (user, otherUser) -> {
                    HashMap<String, User> map = Maps.newHashMap();
                    map.put("user", user);
                    map.put("otherUser", otherUser);
                    return map;
                }
        );
    }

    public DeferredResult<HashMap<String, User>> toDeferredResult(Observable<HashMap<String, User>> details) {
        DeferredResult<HashMap<String, User>> result = new DeferredResult<>();
        // 订阅
```

```
        details.subscribe(new Observer<HashMap<String, User>>() {
            @Override
            public void onCompleted() {
                LOGGER.info("完成...");
            }

            @Override
            public void onError(Throwable throwable) {
                LOGGER.error("发生错误...", throwable);
            }

            @Override
            public void onNext(HashMap<String, User> detail) {
                result.setResult(detail);
            }
        });
        return result;
    }
}
```

这样，代码就编写完成了，不太熟悉RxJava的盆友，建议花一点时间入门RxJava。

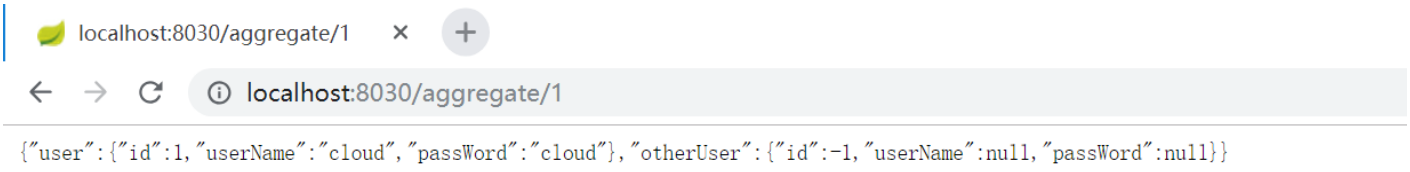
6.测试

(1)启动cloud-discovery-eureka，port=8001

(2)启动cloud-register-user，port=8002

(3)启动cloud-register-gateway-zuul-aggregation，port=8030

(4)访问http://127.0.0.1:8030/aggregate/1，可获得如下结果



由上图可知，本项目即实现了zuul聚合微服务，也实现了Hystrix容错。

扫我有惊喜