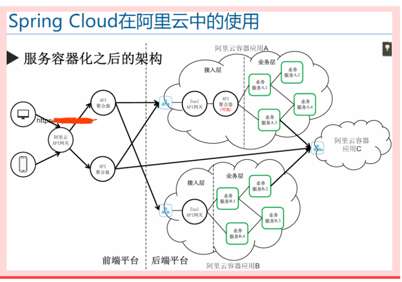
SpringCloud

- 1 微服务概述与SpringCloud
  - 微服务与微服务架构
    - 技术维度理解
      - 微服务化的核心就是将传统的一站式应用，根据业务拆分成一个一个的服务，彻底地去耦合,每一个微服务提供单个业务功能的服务，一个服务做一件事，从技术角度看就是一种小而独立的处理过程，类似进程概念，能够自行单独启动或销毁，拥有自己独立的数据库。
  - 微服务技术栈有哪些
  - SpringCloud是什么
    - 官网说明
    - SpringCloud=分布式微服务架构下的一站式解决方案，是各个微服务架构落地技术的集合体，俗称微服务全家桶
      - All
    - SpringCloud和SpringBoot是什么关系
    - Dubbo是怎么到SpringCloud的?哪些优缺点让你去技术选型
      - 目前成熟的互联网架构（分布式+服务治理Dubbo）
      - 我们把SpringCloud VS DUBBO进行一番对比
      - 总结Cloud与Dubbo
  - 参考资料
    - 官网
      - http://projects.spring.io/spring-cloud/
    - 参考书
      - https://springcloud.cc/spring-cloud-netflix.html
      - 本次开发API说明
        - http://cloud.spring.io/spring-cloud-static/Dalston.SR1/
        - https://springcloud.cc/spring-cloud-dalston.html
      - springcloud中国社区
        - http://springcloud.cn/
      - springcloud中文网
        - https://springcloud.cc/
  - SpringCloud国内使用情况
    - 国内公司

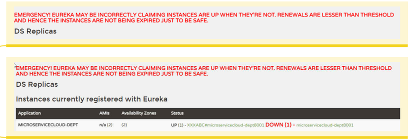
    - 阿里云



- 2 Rest微服务构建案例工程模块
  - 总体介绍
    - 承接着我们的springmvc+mybatis+mysql初级高级课程，以Dept部门模块做一个微服务通用案例Consumer消费者（Client）通过REST调用Provider提供者（Server）提供的服务
    - Maven的分包分模块架构复习
      - 一个Project带着多个Module子模块
      - MicroServiceCloud父工程（Project）下初次带着3个子模块（Module）
        - microservicecloud-api
          - 封装的整体Entity/接口/公共配置等
        - microservicecloud-provider-dept-8001
          - 微服务落地的服务提供者
        - microservicecloud-consumer-dept-80
          - 微服务调用的客户端使用
          - 80端口
    - 直接动手干

  - 本次springCloud版本
  - 构建步骤
    - microservicecloud整体父工程Project
      - 新建父工程microservicecloud，切记是Packageing是pom模式
      - 主要是定义POM文件，将后续各个子模块公用的jar包等统一提出来，类似一个抽象父类
      - POM
    - microservicecloud-api公共子模块Module
      - 新建microservicecloud-api
        - 创建完成后请回到父工程查看pom文件变化
      - 修改POM
      - 新建部门Entity且配合lombok使用
      - mvn clean install后给其它模块引用，达到通用目的。也即需要用到部门实体的话，不用每个工程都定义一份，直接引用本模块即可。
    - microservicecloud-provider-dept-8001 部门微服务提供者Module
      - 1 新建microservicecloud-provider-dept-8001
        - 创建完成后请回到父工程查看pom文件变化
      - 2 POM
      - 3 YML
      - 4 工程src/main/resources目录下新建mybatis文件夹后新建mybatis.cfg.xml文件
      - 5 MySQL创建部门数据库脚本
      - 6 DeptDao部门接口
      - 7 工程src/main/resources/mybatis目录下新建mapper文件夹后再建DeptMapper.xml
      - 8 DeptService部门服务接口
      - 9 DeptServiceImpl部门服务接口实现类
      - 10 DeptController部门微服务提供者REST
      - 11 DeptProvider8001_App主启动类DeptProvider8001_App
      - 12 测试
        - http://localhost:8001/dept/get/2
        - http://localhost:8001/dept/list
      - 13 最终工程展现
    - microservicecloud-consumer-dept-80部门微服务消费者Module
      - 新建microservicecloud-consumer-dept-80
      - POM
      - YML
      - com.atguigu.springcloud.cfgbeans包下ConfigBean的编写（类似spring里面的applicationContext.xml写入的注入Bean）
      - com.atguigu.springcloud.controller包下新建DeptController_Consumer部门微服务消费者REST
        - RestTemplate
          - 是什么
          - 官网及使用
        - 代码
      - DeptConsumer80_App主启动类
      - 测试
        - http://localhost/consumer/dept/get/2
        - http://localhost/consumer/dept/list
        - http://localhost/consumer/dept/add?dname=AI

- 3 Eureka服务注册与发现
  - 是什么
    - Eureka 常用词汇 英 [juˈriːkə]
  - 原理讲解
    - Eureka的基本架构
    - 三大角色
      - Eureka Server 提供服务注册和发现
      - Service Provider服务提供方将自身服务注册到Eureka，从而使服务消费方能够找到
      - Service Consumer服务消费方从Eureka获取注册服务列表，从而能够消费服务
  - 构建步骤
    - microservicecloud-eureka-7001 eureka服务注册中心Module
      - 新建microservicecloud-eureka-7001
      - POM
      - YML
      - EurekaServer7001_App主启动类
        - @EnableEurekaServer
      - 测试
        - http://localhost:7001/
        - 结果页面
          - No application available 没有服务被发现 O(∩_∩)O 因为没有注册服务进来当然不可能有服务被发现
    - microservicecloud-provider-dept-8001 将已有的部门微服务注册进eureka服务中心
      - 修改microservicecloud-provider-dept-8001
      - POM
        - 修改部分
        - 完整内容
      - YML
        - 修改部分
        - 完整内容
      - DeptProvider8001_App主启动类
        - @EnableEurekaClient
      - 测试
        - 先要启动EurekaServer
        - http://localhost:7001/
        - 微服务注册名配置说明
    - actuator与注册微服务信息完善
      - 主机名称:服务名称修改
        - 当前问题
        - 修改microservicecloud-provider-dept-8001
          - YML
            - 修改部分
            - 完整内容
        - 修改之后
      - 访问信息有IP信息提示
        - 当前问题
          - 没有IP提示
        - 修改microservicecloud-provider-dept-8001
          - YML
            - 修改部分
            - 完整内容
        - 修改之后
      - 微服务info内容详细信息
        - 当前问题
          - 超链接点击服务报告ErrorPage
        - 修改microservicecloud-provider-dept-8001
          - POM
            - 修改部分
            - 完整内容
        - 总的父工程microservicecloud修改pom.xml添加构建build信息
          - POM
            - 修改部分
            - 完整内容
        - 修改microservicecloud-provider-dept-8001
          - YML
            - 修改部分
            - 完整内容
    - eureka自我保护
      - 演示Case
      - 故障现象



      - 导致原因
        - 一句话：某时刻某一个微服务不可用了，eureka不会立刻清理，依旧会对该微服务的信息进行保存
  - 集群配置
    - 原理说明
    - 新建microservicecloud-eureka-7002/microservicecloud-eureka-7003
    - 按照7001为模板粘贴POM
    - 修改7002和7003的主启动类
    - 修改映射配置
      - 找到C:\Windows\System32\drivers\etc路径下的hosts文件
      - 修改映射配置添加进hosts文件
        - 127.0.0.1 eureka7001.com
        - 127.0.0.1 eureka7002.com
        - 127.0.0.1 eureka7003.com
    - 3台eureka服务器的yml配置
      - 7001
      - 7002
      - 7003
    - microservicecloud-provider-dept-8001微服务发布到上面3台eureka集群配置中
      - YML

- 4 Hystrix断路器
  - 概述
    - 分布式系统面临的问题
    - 是什么
    - 能干嘛
      - 服务降级
      - 服务熔断
      - 服务限流
      - 接近实时的监控
      - 。。。。。。
    - 官网资料
      - https://github.com/Netflix/Hystrix/wiki/How-To-Use
  - 服务熔断
    - 是什么
    - 参考microservicecloud-provider-dept-8001
    - 新建microservicecloud-provider-dept-hystrix-8001
    - POM
      - 修改内容
      - 全部内容
    - YML
    - 修改DeptController
      - @HystrixCommand报异常后如何处理
        - 一旦调用服务方法失败并抛出了错误信息后，会自动调用@HystrixCommand标注好的fallbackMethod调用类中的指定方法
      - 代码内容
    - 修改主启动类DeptProvider8001_Hystrix_App并添加新注解@EnableCircuitBreaker
    - 测试
      - 3个eureka先启动
      - 主启动类DeptProvider8001_Hystrix_App
      - Consumer启动microservicecloud-consumer-dept-80
      - http://localhost/consumer/dept/get/112
  - 服务降级
    - 是什么
    - 服务降级处理是在客户端实现完成的，与服务端没有关系
    - 修改microservicecloud-api工程，根据已经有的DeptClientService接口新建一个实现了FallbackFactory接口的类DeptClientServiceFallbackFactory
      - 千万不要忘记在类上面新增@Component注解，大坑！！！
    - 修改microservicecloud-api工程，DeptClientService接口在注解@FeignClient中添加fallbackFactory属性值
    - microservicecloud-api工程
      - mvn clean install
    - microservicecloud-consumer-dept-feign工程修改YML
      - YML
    - 测试
      - 3个eureka先启动
      - 微服务microservicecloud-provider-dept-8001启动
      - microservicecloud-consumer-dept-feign启动
      - 正常访问测试
        - http://localhost/consumer/dept/get/1
      - 故意关闭微服务microservicecloud-provider-dept-8001
      - 客户端自己调用提示
        - http://localhost/consumer/dept/get/1
        - 此时服务端provider已经down了，但是我们做了服务降级处理，让客户端在服务端不可用时也会获得提示信息而不会挂起耗死服务器
  - 服务监控hystrixDashboard
    - 概述
    - Case步骤
      - 新建工程microservicecloud-consumer-hystrix-dashboard
      - POM
        - 修改内容
        - 全部内容
      - YML
      - 主启动类改名+新注解@EnableHystrixDashboard
      - 所有Provider微服务提供类(8001/8002/8003)都需要监控依赖配置
      - 启动microservicecloud-consumer-hystrix-dashboard该微服务监控消费端
        - http://localhost:9001/hystrix
      - 启动3个eureka集群
      - 启动microservicecloud-provider-dept-hystrix-8001
        - http://localhost:8001/dept/get/1
        - http://localhost:8001/hystrix.stream
      - 启动的相关微服务工程
      - 监控测试
        - 多次刷新http://localhost:8001/dept/get/1
        - 观察监控窗口
          - 填写监控地址
          - 监控结果
          - 如何看?
            - 7色
            - 1圈
            - 1线
            - 整图说明
          - 搞懂一个才能看懂复杂的

- 5 Ribbon负载均衡
  - 概述
    - 是什么
    - 能干吗
      - LB（负载均衡）
        - 集中式LB
        - 进程内LB
    - 官网资料
      - https://github.com/Netflix/ribbon/wiki/Getting-Started
  - Ribbon配置初步
    - 修改microservicecloud-consumer-dept-80工程
    - 修改pom.xml文件
      - 内容
    - 修改application.yml 追加eureka的服务注册地址
      - 内容
    - 对ConfigBean进行新注解@LoadBalanced 获得Rest时加入Ribbon的配置
    - 主启动类DeptConsumer80_App添加@EnableEurekaClient
    - 修改DeptController_Consumer客户端访问类
    - 先启动3个eureka集群后，再启动microservicecloud-provider-dept-8001并注册进eureka
    - 启动microservicecloud-consumer-dept-80
    - 测试
      - http://localhost/consumer/dept/get/1
      - http://localhost/consumer/dept/list
      - http://localhost/consumer/dept/add?dname=大数据部
    - 小总结
      - Ribbon和Eureka整合后Consumer可以直接调用服务而不用再关心地址和端口号
  - Ribbon负载均衡
    - 架构说明
    - 参考microservicecloud-provider-dept-8001，新建两份，分别命名为8002，8003
    - 新建8002/8003数据库，各自微服务分别连各自的数据库
      - 8002SQL脚本
      - 8003SQL脚本
    - 修改8002/8003各自YML
      - 8002YML
      - 8003YML
      - 备注
        - 端口
        - 数据库链接
          - url: jdbc:mysql://localhost:3306/cloudDB01
        - 对外暴露的统一的服务实例名
          - spring: application: name: microservicecloud-dept （8001/8002/8003三个名字都要一样）
    - 启动3个eureka集群配置区
    - 启动3个Dept微服务启动并各自测试通过
      - http://localhost:8001/dept/list
      - http://localhost:8002/dept/list
      - http://localhost:8003/dept/list
    - 启动microservicecloud-consumer-dept-80
    - 客户端通过Ribbo完成负载均衡并访问上一步的Dept微服务
      - http://localhost/consumer/dept/list
      - 注意观察看到返回的数据库名字，各不相同，负载均衡实现
    - 总结：Ribbon其实就是一个软负载均衡的客户端组件，他可以和其他所需请求的客户端结合使用，和eureka结合只是其中的一个实例。

- 6 Feign负载均衡
  - 概述
    - Feign 常用词汇 英 [feɪn] 美 [feɪn] vt. 假装；捏造 vi. 假装；装作
  - Feign使用步骤
    - 参考microservicecloud-consumer-dept-80
    - 新建microservicecloud-consumer-dept-feign
      - 修改主启动类名字
        - DeptConsumer80_Feign_App
    - microservicecloud-consumer-dept-feign工程pom.xml修改，主要添加对feign的支持
    - 修改microservicecloud-api工程
      - POM
      - 新建DeptClientService接口并新增注解@FeignClient
        - @FeignClient
      - mvn clean
      - mvn install
    - microservicecloud-consumer-dept-feign工程修改Controller，添加上一步新建的DeptClientService接口
    - microservicecloud-consumer-dept-feign工程修改主启动类
      - @EnableFeignClients
    - 测试
      - 启动3个eureka集群
      - 启动3个部门微服务8001/8002/8003
      - 启动Feign自己启动
      - http://localhost/consumer/dept/list
      - Feign自带负载均衡配置项
    - 小总结

- 7 zuul路由网关
  - 概述
    - 是什么
    - 能干嘛
      - 路由
      - 过滤
    - 官网资料
      - https://github.com/Netflix/zuul/wiki/Getting-Started
  - 路由基本配置
    - 新建Module模块microservicecloud-zuul-gateway-9527
    - POM
      - 修改内容
      - 全部内容
    - YML
    - hosts修改
      - 127.0.0.1 myzuul.com
    - 主启动类
      - @EnableZuulProxy
    - 启动
      - 三个eureka集群
      - 一个服务提供类microservicecloud-provider-dept-8001
      - 一个路由
    - 测试
      - 不用路由
        - http://localhost:8001/dept/get/2
      - 启用路由
        - http://myzuul.com:9527/microservicecloud-dept/dept/get/2
  - 路由访问映射规则
    - 工程microservicecloud-zuul-gateway-9527
    - 代理名称
      - YML
      - 此时问题
        - 路由访问OK
          - http://myzuul.com:9527/mydept/dept/get/1
        - 原路径访问OK
          - http://myzuul.com:9527/microservicecloud-dept/dept/get/2
    - 原真实服务名忽略
      - YML
      - 单个具体，多个可以用"*"
    - 设置统一公共前缀
      - YML
      - http://myzuul.com:9527/atguigu/mydept/dept/get/1
    - 最后YML

Presented with XMind