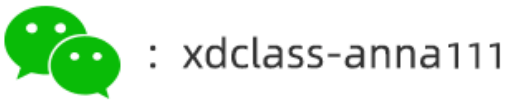


愿景："让编程不再难学，让技术与生活更加有趣"

更多架构课程请访问 xdclass.net

第一章 新一代微服务全家桶 Alibaba Cloud+Docker容器化零基础到实战介绍

第1集 新一代微服务AlibabaCloud 全家桶 +Docker课程介绍

简介：讲解新一代微服务AlibabaCloud课程介绍，课程适合人员和学后水平

- **课程介绍 + 学后水平**
  - 新一代微服务全家桶 Alibaba Cloud+Docker容器化零基础到实战课程 是对中高级后端工程师而定制的，分为多个大方向
  - **首发 SpringCloud Hoxton.SR8 + AlibabaCloud 2.2.1 + JDK11 专题课程**
    - 从零基础讲解分布式微服务架构，业界常用微服务解决方案
    - 基于小滴课堂在线教育架构拆分微服务，Maven聚合工程 + Mybatis操作数据库
    - 零基础安装服务治理Nacos+AlibabaCloud服务连接
    - 从0教你掌握JDK的LTS版本 - JDK11升级和里面的坑
    - 深入源码讲解 RPC调用 Ribbon+负载均衡策略实现
    - 玩转新一代Open-Feign组件+多种HTTP调用案例
    - 分布式面试题 CAP + BASE理论，注册中心架构选择
    - 掌握新一代微服务容错组件Alibaba-Sentinel多种流控策略+熔断降级实操
    - 深入掌握新一代SpringCloud-Gateway+多种路由断言+自定义全局鉴权过滤器
    - 玩转分布式链路追踪Zipkin+Sleuth实战 + Mysql数

据持久

- 掌握分布式配置中心Nacos实战+动态配置下发+新版本里面的坑
- 零基础掌握Docker容器在阿里云ECS服务器安装+私有镜像仓库搭建
- 阿里云Docker部署Dockerfile+JDK11+AlibabaCloud全家桶镜像构建推送
- 阿里云Docker部署Sentinel+Nacos+Zipkin+Gateway +AlibabaCloud全家桶

注意：我们专题课程并非大杂烩，网上有些视频几百集，但是微服务只讲了一丁点且采用老版本+误导

比如有些用老版本 Spring Cloud的SR7 + JDK9， 这个在生产环境是绝对不给用的版本

理由：一定不能用JDK9版本，功能性版本官方只维护半年，目前Oracle官方早就不维护JDK9了，是否有漏洞或者BUG被黑客利用都不得而知，且不同版本各个组件兼容问题很多

全职讲师和企业讲师的区别就在这里，全职讲师脱离当下互联网公司实际采用技术选择和缺乏大厂前言技术视角

- 为什么要学为AlibabaCloud课程
  - 互联网公司业务多数是微服务架构，可以快速实现扩容，独立部署，故障和资源的隔离性等等
  - 招聘需求

```
https://www.zhipin.com/job_detail/ee4a69bcdce2ff6633Jz2Ny4E1I~.html

https://www.zhipin.com/job_detail/9ffe5eecef4dbc8c33d70966E1M~.html
```

- 适合人群
  - 中高级后端工程师、项目经理、CTO 更新必备技术栈
  - 从传统软件公司过渡到互联网公司的人员

    - 全栈工程师、运维工程师

- 课程技术技术栈和环境说明
    - SpringBoot.2.3.3 + Mybatis+ SpringCloud Hoxton.SR8+ AlibabaCloud 2.2.1
    - JDK8 + JDK11 + IDEA旗舰版 + Docker1.13 + 阿里云 CentOS7.X

- 学习形式
    - 视频讲解 + 文字笔记 + 原理分析 + 交互流程图
    - 配套源码 + 笔记 + 技术群答疑(我答疑，联系客服进群)
    - 课程有配套的源码，在每章-每集的资料里面，如果没用到代码就不保存
    - 只要是我的课程和项目-我会一直维护下去，大家不用担心！！！
- 不要贪求多而复杂，先学会走路，再学跑步，把这个课程学完足够开发和部署多数公司微服务项目

从零到一搭建微服务全家桶组件，基于旭瑶&小滴课堂在线教育项目架构

微服务面试题+源码分析+高可用架构+新版本的坑

阿里云Docker容器化部署AlibabaCloud全家桶

不增加复杂业务逻辑，让大家更好掌握AlibabaCloud全家桶

我们有成套路线，从零到高级工程师学习视频，不止有高级源码分析，还有综合项目实战

第2集 微服务AlibabaCloud全家桶课程大纲速览和效果演示

简介：讲解AlibabaCloud全家桶课程大纲和效果演示

- 线上部署效果演示
- 课程学前基础
  - SpringBoot2.x基础
  - Linux基础
  - PS:不会上面的基础也没关系,这些基础都有课程，联系我们客服即可，且是刚录制的新版课程
- 目录大纲浏览

- 学习要求

- 保持谦虚好学、术业有专攻、在校的学生，有些知识掌握也比工作好几年掌握的人厉害
- 课程有配套的源码，在每章-每集的资料里面，如果没用到代码就不保存，目录介绍
- 如果自己操作的情况和视频不一样，导入课程代码对比验证基本就可以发现问题了
- 官方要求，务必保持版本一致，不然会遇到很多问题。

愿景："让编程不再难学，让技术与生活更加有趣"

更多架构课程请访问 xdclass.net

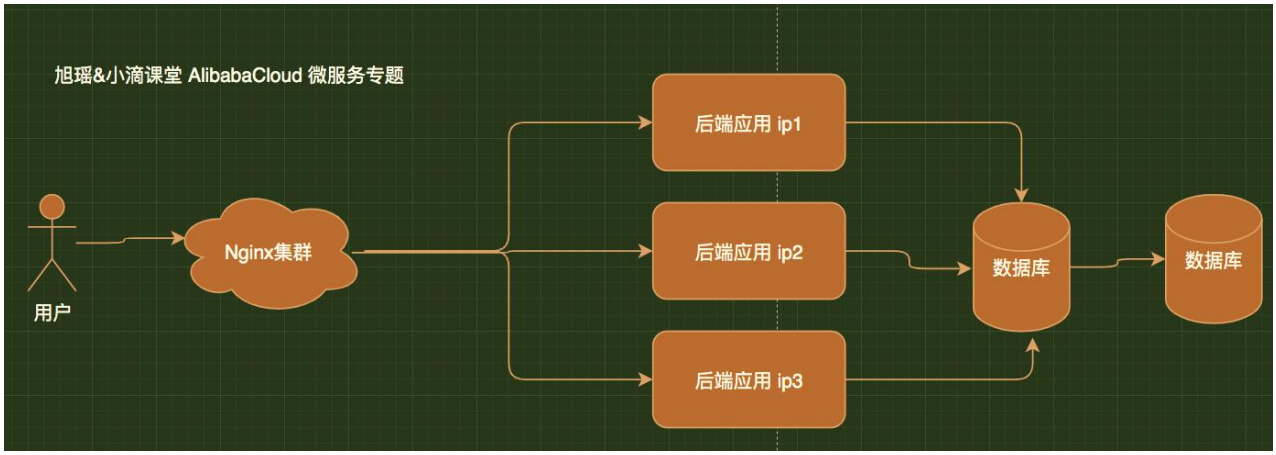
第二章 单机应用到分布式架构演进核心知识

第1集 传统单机-分布式架构演进历史

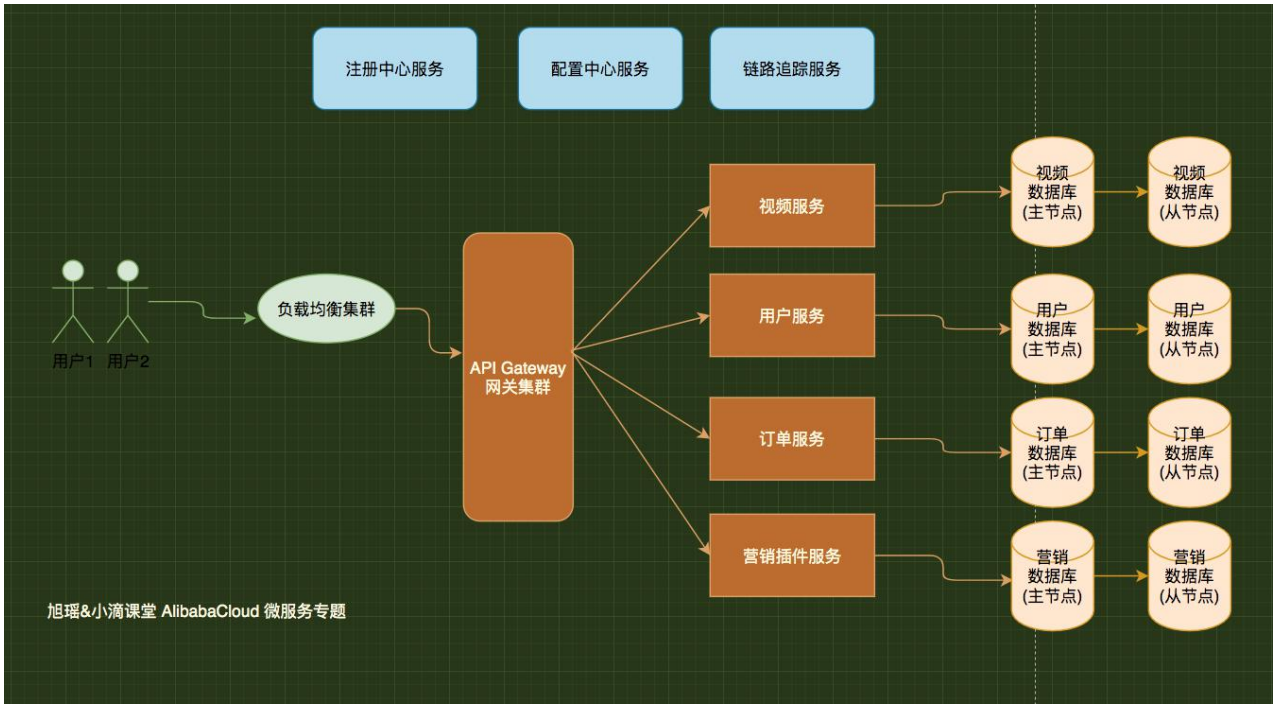
简介：讲解单机到微服务架构的演进

- 单机架构
  - 优点： 易于测试 便于集成 小型项目友好

- 缺点： 开发速度慢 启动时间长 依赖庞大



- 分布式架构
  - SOA： Service Oriented Architecture 面向服务的架构 其中包含多个服务， 服务之间通过相互依赖最终提供一系列的功能, 一个服务 通常以独立的形式存在与操作系统进程中, 各个服务之间 通过网络调用。
  - 微服务：将一个大的单体应用进行细粒度的服务化拆分，每个拆分出来的服务各自独立打包部署，各个服务之间 通过网络调用。
  - 优点
    - 易开发、理解和维护
    - 独立的部署和启动
  - 缺点
    - 分布式系统-》分布式事务问题
    - 需要管理多个服务-》服务治理



第2集 大话微服务全家桶架构组成

简介：微服务架构常见的核心组件

- 常见组件
  - 网关

        - 路由转发 + 过滤器
            - /api/v1/video/ 视频服务
            - /api/v1/order/ 订单服务
            - /api/v1/user/ 用户服务

    - 服务发现注册
        - 调用和被调用方的信息维护

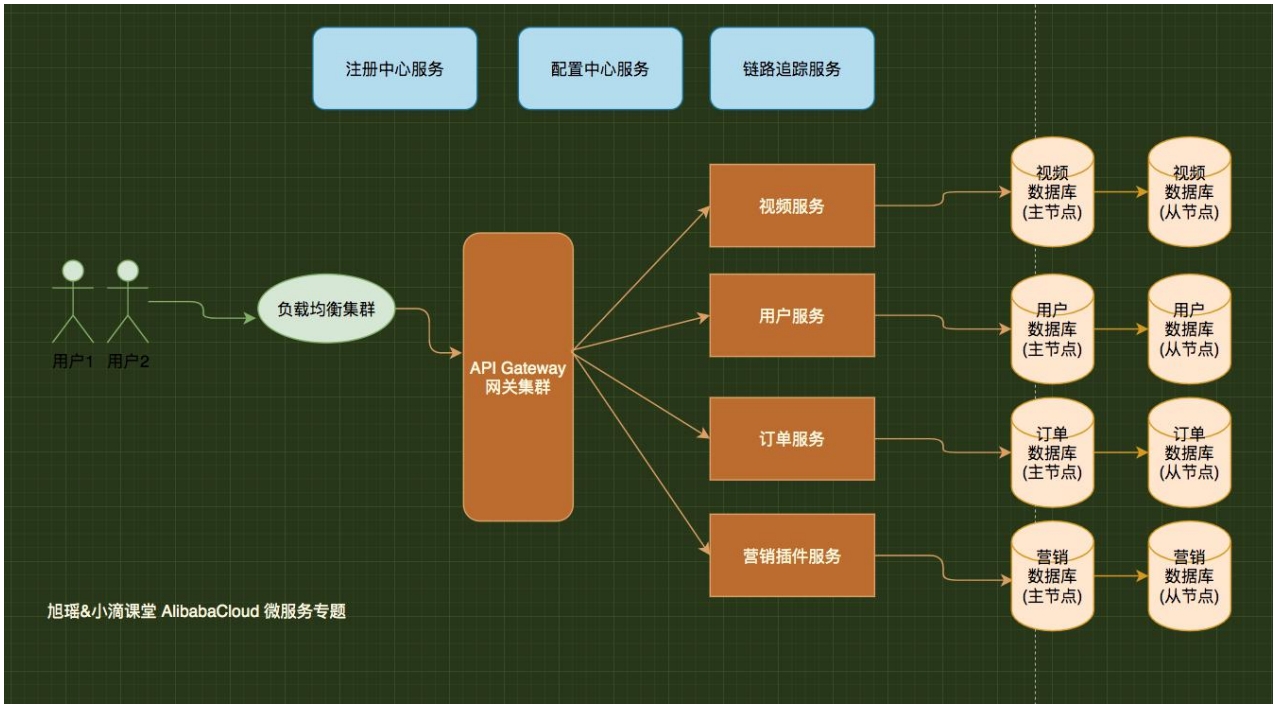
    - 配置中心
        - 管理配置，动态更新 application.properties

    - 链路追踪
        - 分析调用链路耗时 例子：下单-》查询商品服务获取商品价格-》查询用户信息-》保存数据库

    - 负载均衡器
        - 分发流量到多个节点，降低压力

    - 熔断
        - 保护自己和被调用方

- 微服务核心组件图


注册中心服务
配置中心服务
链路追踪服务
用户1
用户2
负载均衡集群
API Gateway
网关集群
视频服务
用户服务
订单服务
营销插件服务
视频
数据库
(主节点)
视频
数据库
(从节点)
用户
数据库
(主节点)
用户
数据库
(从节点)
订单
数据库
(主节点)
订单
数据库
(从节点)
营销
数据库
(主节点)
营销
数据库
(从节点)
旭瑶&小滴课堂 AlibabaCloud 微服务专题

第3集 业界微服务架构常见解决方案

简介：讲解业务微服务架构常见解决方案

- ServiceComb
  - 华为内部的CSE(Cloud Service Engine)框架开源, 一个微服务的开源解决方案,社区相对于下面几个比较小
  - 文档不多，通信领域比较强

- dubbo
  - zookeeper + dubbo + springmvc/springboot
  - 官方地址：http://dubbo.apache.org/#!/?lang=zh-cn
  - 配套
    - 通信方式：rpc
    - 注册中心：zookeper/redis/nacos
    - 配置中心：diamond、nacos

- SpringCloud
  - 全家桶+轻松嵌入第三方组件(Netflix 奈飞)
  - 官网：https://spring.io/projects/spring-cloud
  - 配套

        - 通信方式：http restful
        - 注册中心：eruka
        - 配置中心：config
        - 断路器：hystrix
        - 网关：zuul/gateway
        - 分布式追踪系统：sleuth+zipkin

- Spring Alibaba Cloud
    - 全家桶+阿里生态多个组件组合+SpringCloud支持
    - 官网 https://spring.io/projects/spring-cloud-alibaba
    - 配套
        - 通信方式：http restful
        - 注册中心：nacos
        - 配置中心：nacos
        - 断路器：sentinel
        - 网关：gateway
        - 分布式追踪系统：sleuth+zipkin

旭瑶&小滴课堂 愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第三章 新一代微服务AlibabaCloud介绍和架构环境准备

第1集 新一代微服务架构AlibabaCloud全家桶介绍

简介：讲解AlibabaCloud核心组件介绍

- 官网介绍
  - https://spring.io/projects/spring-cloud-alibaba#overview

- 为什么要选择AlibabaCloud , 和SpringCloud的区别
- SpringCloud和AlibabaCloud组件存在很大交集，互相配合
    - SpringCloud很多组件是基于第三方整合，目前多个已经不更新了，比如zuul、eureka、hystrix等
    - AlibabaCloud 提供一站式微服务解决方法，已经和SpringCloud进行了整合，组件互相支持

- AlibabaCloud全家桶介绍
    - https://github.com/alibaba/spring-cloud-alibaba
    - 服务注册发现：Nacos
    - 服务限流降级：Sentinel
    - 分布配置中心：Nacos
    - 服务网关：SpringCloud Gateway
    - 服务之间调用：Feign、Ribbon
    - 链路追踪：Sleuth+Zipkin

- 版本说明
    - 【注意文档：】官方经常改地址，如果本课程的地址失效后重新搜索下找入口
    - Spring5以上

- SpringBoot2.x以上
- AlibabaCloud 版本 2.2.x https://spring.io/projects/spring-cloud-alibaba#learn
- SpirngCloud版本 Hoxton https://spring.io/projects/spring-cloud

Release Train	Boot Version
Hoxton	2.2.x, 2.3.x (Starting with SR5)
Greenwich	2.1.x
Finchley	2.0.x
Edgware	1.5.x
Dalston	1.5.x

第2集 小滴课堂在线教育微服务模块设计和环境准备

简介：在线教育环微服务模块划分和环境准备

- 在线教育模块划分
  - 视频服务
  - 订单服务
  - 用户服务
- 必备基础环境：JDK8以上版本+Maven3.5(采用默认)+IDEA旗舰版+Mysql5.7以上版本
- 操作系统：Linux Centos7 64位(虚拟机) 或者 Mac苹果系统
  - 虚拟机可以搜索博文
  - Windows有些软件会不兼容，且坑难排查
  - 学习期间务必关闭防火墙
- 课程围绕这个基础项目进行学习 小而精的方式学习微服务

第3集 小滴课堂在线教育微服务数据库介绍和数据导入

简介：讲解小滴课堂在线教育数据库表介绍和导入

- 采用3个数据库，每个服务单独一个库

- 视频服务数据库 video表

```
CREATE TABLE `video` (
  `id` int(11) unsigned NOT NULL
AUTO_INCREMENT,
  `title` varchar(524) DEFAULT NULL COMMENT '视
频标题',
  `summary` varchar(1026) DEFAULT NULL COMMENT
'概述',
  `cover_img` varchar(524) DEFAULT NULL COMMENT
'封面图',
  `price` int(11) DEFAULT NULL COMMENT '价格,
分',
  `create_time` datetime DEFAULT NULL COMMENT
'创建时间',
  `point` double(11,2) DEFAULT '8.70' COMMENT
'默认8.7，最高10分',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=48 DEFAULT
CHARSET=utf8;
```

```
INSERT INTO `video` (`id`, `title`, `summary`,
`cover_img`, `price`, `create_time`, `point`)
VALUES
```

```
  (30, '互联网架构之JAVA虚拟机JVM零基础到高级实战',
'https://xdvideo-file.oss-cn-
shenzhen.aliyuncs.com/video/2020/maven/%E8%AF%A
6%E6%83%85%E5%9B%BE.png', 'https://xdvideo-
file.oss-cn-
shenzhen.aliyuncs.com/video/2020/maven/%E5%AE%9
8%E7%BD%91%E4%B8%BB%E5%9B%BE-mawen.png', 3980,
'2021-06-24 22:14:00', 9.10),
  (40, '全新微信小程序零基础到项目实战',
'https://xdvideo-file.oss-cn-
shenzhen.aliyuncs.com/video/2020/%E5%BE%AE%E4%B
F%A1%E5%B0%8F%E7%A8%8B%E5%BA%8F/%E8%AF%A6%E6%83
%85%E5%9B%BE.png', 'https://xdvideo-file.oss-
cn-
shenzhen.aliyuncs.com/video/2020/%E5%BE%AE%E4%B
F%A1%E5%B0%8F%E7%A8%8B%E5%BA%8F/%E5%AE%98%E7%BD
%91%E4%B8%BB%E5%9B%BE-
%E5%B0%8F%E7%A8%8B%E5%BA%8F.png', 5980, '2021-
01-18 22:14:00', 9.10),
  (41, '玩转搜索框架ElasticSearch7.x实战',
'https://xd-video-pc-img.oss-cn-
beijing.aliyuncs.com/xdclass_pro/video/2019_bac
kend/elasticsearch7_detail.jpeg', 'https://xd-
video-pc-img.oss-cn-
beijing.aliyuncs.com/xdclass_pro/video/2019_bac
kend/elasticsearch7.png', 4880, '2021-01-10
22:14:00', 8.70),
```

```
  (45, 'Docker实战视频教程入门到高级
dockerfile/compose-Harbor', 'https://xdvideo-
file.oss-cn-
shenzhen.aliyuncs.com/video/2020/Docker/%E8%AF%
A6%E6%83%85%E5%9B%BE.jpeg', 'https://xdvideo-
file.oss-cn-
shenzhen.aliyuncs.com/video/2020/Docker/%E5%AE%
98%E7%BD%91%E4%B8%BB%E5%9B%BE-docker.png',
5980, '2021-01-10 22:14:00', 9.30),
  (46, '新版javase零基础到高级教程小白自学编程',
'https://xdvideo-file.oss-cn-
shenzhen.aliyuncs.com/video/2020/%E6%96%B0%E7%8
9%88javase/%E8%AF%A6%E6%83%85%E5%9B%BE.png',
'https://file.xdclass.net/video/2020/%E6%96%B0%
E7%89%88javase/%E5%AE%98%E7%BD%91%E4%B8%BB%E5%9
B%BE-javase.png', 3980, '2021-01-24 22:14:00',
8.80),
  (47, 'Nodejs教程零基础入门到项目实战前端视频教程',
'https://xdvideo-file.oss-cn-
shenzhen.aliyuncs.com/video/2020/node/%E5%AE%98
%E7%BD%91%E8%AF%A6%E6%83%85%E5%9B%BE-node.png',
'https://xdvideo-file.oss-cn-
shenzhen.aliyuncs.com/video/2020/node/%E5%AE%98
%E7%BD%91%E4%B8%BB%E5%9B%BE-node.png', 6980,
'2021-01-24 22:14:00', 8.90);
```

- 用户服务数据库 user表

```
CREATE TABLE `user` (
  `id` int(11) unsigned NOT NULL
AUTO_INCREMENT,
  `phone` varchar(32) DEFAULT NULL,
  `pwd` varchar(128) DEFAULT NULL,
  `sex` int(2) DEFAULT NULL,
  `img` varchar(128) DEFAULT NULL,
  `create_time` datetime DEFAULT NULL,
  `role` int(11) DEFAULT NULL COMMENT '1是普通用
户，2是管理员',
  `username` varchar(128) DEFAULT NULL,
  `wechat` varchar(128) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT
CHARSET=utf8mb4;
```

```
INSERT INTO `user` (`id`, `phone`, `pwd`,
`sex`, `img`, `create_time`, `role`,
`username`, `wechat`)
VALUES
  (1, '123', '666', 1, 'xdclass.net', '2021-09-
09 00:00:00', 1, 'jack', 'xdclass6'),
  (2, '2323432', '794666918', 1, 'wwwww',
'2020-05-20 04:54:01', 1, '小滴Anna姐姐',
'xdclass-anna'),
  (3, '2323432', 'xdclass-lw', 1, 'wwwww',
'2020-05-20 04:54:42', 1, '二当家小D',
'xdclass1'),
  (4, '2323432', '3232323', 1, 'wwwww', '2020-
05-20 04:55:07', 1, '老王', 'xdclass-lw');
```

- 订单服务数据库video_order表

```
CREATE TABLE `video_order` (
  `id` int(11) unsigned NOT NULL
AUTO_INCREMENT,
  `out_trade_no` varchar(64) DEFAULT NULL
COMMENT '订单唯一标识',
  `state` int(11) DEFAULT NULL COMMENT '0表示未
支付，1表示已支付',
  `create_time` datetime DEFAULT NULL COMMENT
'订单生成时间',
  `total_fee` int(11) DEFAULT NULL COMMENT '支付
金额，单位分',
  `video_id` int(11) DEFAULT NULL COMMENT '视频
主键',
  `video_title` varchar(256) DEFAULT NULL
COMMENT '视频标题',
  `video_img` varchar(256) DEFAULT NULL COMMENT
'视频图片',
  `user_id` int(12) DEFAULT NULL COMMENT '用户
id',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=42 DEFAULT
CHARSET=utf8;
```

第4集 Maven聚合工程创建微服务项目实战

简介：使用Maven聚合工程创建微服务架构

- maven聚合工程
  - xdclass-common
  - xdclass-video-service
  - xdclass-user-service
  - xdclass-order-service

- 创建聚合工程（记得删除聚合工程src目录）

```
<modelVersion>4.0.0</modelVersion>

    <groupId>net.xdclass</groupId>
    <artifactId>cloud-dmeo</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modules>
        <module>xdclass-cloud-common</module>
        <module>xdclass-cloud-user</module>
        <module>xdclass-cloud-video</module>
```

```
        <module>xdclass-cloud-order</module>
    </modules>

    <!-- 一般来说父级项目的packaging都为pom,
packaging默认类型jar类型-->
    <packaging>pom</packaging>

    <properties>
        <java.version>1.8</java.version>
    </properties>


    <dependencyManagement>
        <dependencies>
            <!--
https://mvnrepository.com/artifact/org.spring
framework.boot/spring-boot-
dependencies/2.3.3.RELEASE-->
            <dependency>

<groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-
dependencies</artifactId>

<version>2.3.3.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
```

```
            <!--
https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-dependencies/Hoxton.SR8-->
            <dependency>

<groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>Hoxton.SR8</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>

            <!--
https://mvnrepository.com/artifact/com.alibaba.cloud/spring-cloud-alibaba-dependencies/2.2.1.RELEASE-->
            <dependency>

<groupId>com.alibaba.cloud</groupId>
                <artifactId>spring-cloud-alibaba-dependencies</artifactId>

<version>2.2.1.RELEASE</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
```

```
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>

<groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-
maven-plugin</artifactId>
                <configuration>
                    <fork>true</fork>

<addResources>true</addResources>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

- 创建3个子项目

- 添加子项目依赖

```
 <dependencies>
        <dependency>

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

        <dependency>
            <groupId>net.xdclass</groupId>
            <artifactId>xdclass-
common</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

- 注意: 有些包maven下载慢，等待下载如果失败
  - 删除本地仓库spring相关的包，重新执行 mvn install

第5集 AlibabaCloud微服务Mybatis连接Mysql数据库

简介：微服务打通Mybatis连接Mysql数据库

- 创建common包实体类

```
public class User {
        private Integer id;
        private String name;
        private String pwd;
        private String headImg;
        private String phone;
        private Date createTime;
        private String wechat;
}

public class Video {
    private Integer id;
    private String title;
    private String summary;
    private String coverImg;
    private Integer  price;
    private Date createTime;
```

```
    private Double point;
}

public class VideoOrder {
    private Integer id;
    private String outTradeNo;
    private Integer state;
    private Date createTime;
    private  Integer totalFee;
    private Integer videoId;
    private String videoTitle;
    private String videoImg;
    private Integer userId;
}
```

- Mybatis依赖导入+数据库配置
  - xdclass-video-service
  - xdclass-user-service
  - xdclass-order-service

- 聚合工程pom.xml修改【注意】

```
    <properties>
        <java.version>1.8</java.version>
```

```
<maven.compiler.source>1.8</maven.compiler.so
urce>

<maven.compiler.target>1.8</maven.compiler.ta
rget>
    </properties>


  <dependency>

<groupId>org.mybatis.spring.boot</groupId>
    <artifactId>mybatis-spring-boot-
starter</artifactId>
    <version>2.1.2</version>
    <type>pom</type>
    <scope>import</scope>
  </dependency>
```

- 添加mybatis依赖和数据库驱动

```
      <dependency>

<groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-
starter</artifactId>
        </dependency>

        <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-
java</artifactId>
      </dependency>
```

- 3个模块配置数据库连接（记得修改 端口、应用名称、数据库名称）

```
server:
  port: 9000

spring:
  application:
    name: xdclass-video-service
  datasource:
    driver-class-name:
com.mysql.cj.jdbc.Driver
```

```
    url:
jdbc:mysql://127.0.0.1:3306/cloud_video?
useUnicode=true&characterEncoding=utf-
8&useSSL=false
    username: root
    password: xdclass.net

# 控制台输出sql、下划线转驼峰
mybatis:
  configuration:
    log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
```

- controller->service->mapper 开发
- application配置

```
@SpringBootApplication
@MapperScan("net.xdclass.dao")
```

第6集 初探微服务之间的调用-下单购买视频

简介：初探微服务直接的调用-下单购买视频

- 服务直接怎么调用：

```
RPC：
  远程过程调用，像调用本地服务(方法)一样调用服务器的服务
  支持同步、异步调用
  客户端和服务器之间建立TCP连接，可以一次建立一个，也可以多个调用复用一次链接
  RPC数据包小
    protobuf
    thrift
  rpc：编解码，序列化，链接，丢包，协议

Rest(Http)：
  http请求，支持多种协议和功能
  开发方便成本低
  http数据包大
  java开发：resttemplate或者httpclient
```

- 用户下单
  - 订单服务->视频服务(查询价格和冗余信息)

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

Video video =
restTemplate.getForObject("http://localhost:9
000/api/v1/video/find_by_id?
videoId="+videoId,Video.class);
```

- postman工具测试(安装包在资料里面)

- 存在的问题：
  - 服务之间的IP信息写死
  - 服务之间无法提供负载均衡
  - 多个服务直接关系调用维护复杂

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第四章 AlibabaCloud核心组件服务治理Nacos实战

第1集 什么是微服务的注册中心

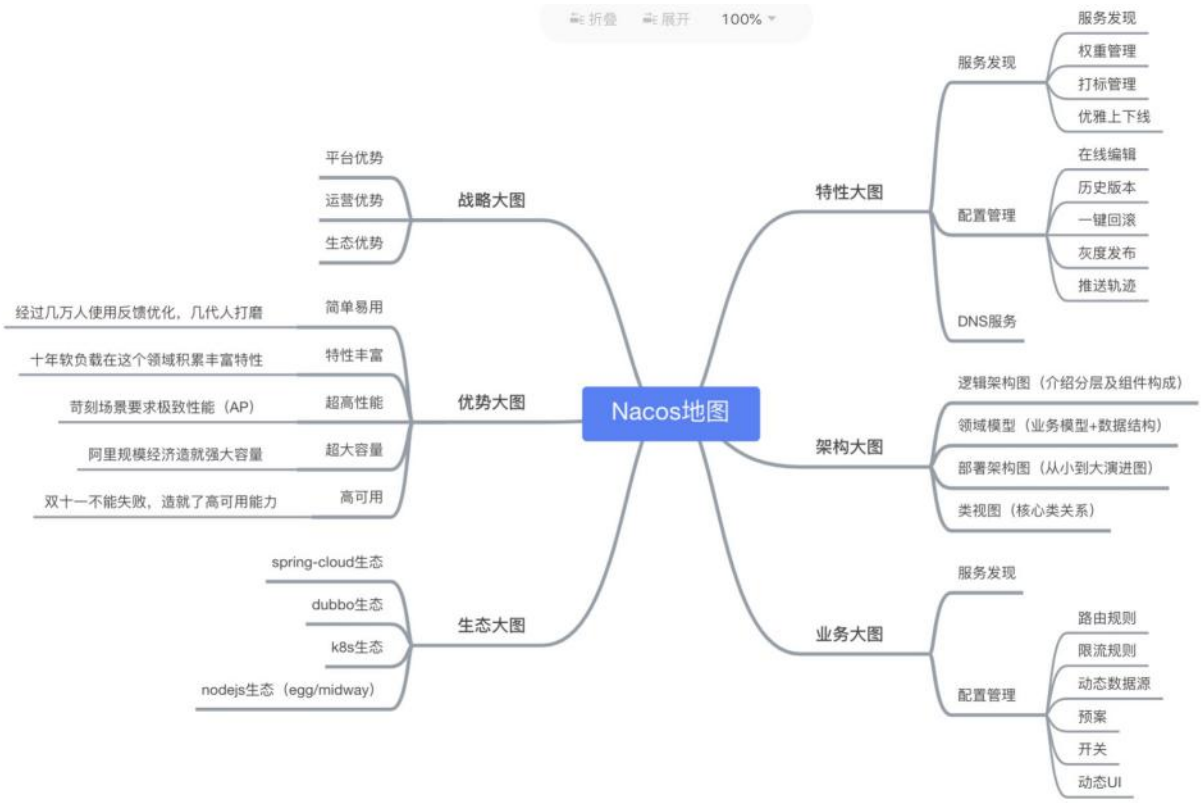
简介：什么是注册中心和常见的注册中心有哪些

- 微服务架构图讲解
- 什么是注册中心（服务治理）

    - 服务注册：服务提供者provider，启动的时候向注册中心上报自己的网络信息
        - 服务发现：服务消费者consumer,启动的时候向注册中心上报自己的网络信息，拉取provider的相关网络信息
    - 核心:服务管理,是有个服务注册表，心跳机制动态维护，服务实例在启动时注册到服务注册表，并在关闭时注销。

- 为什么要用
    - 微服务应用和机器越来越多，调用方需要知道接口的网络地址，如果靠配置文件的方式去控制网络地址，对于动态新增机器，维护带来很大问题

- 主流的注册中心：zookeeper、Eureka、consul、etcd、Nacos
    - AlibabaCloud搭配最好的是Nacos，且服务的注册发现之外，还支持动态配置服务
    - 参考图片(nacos官网)

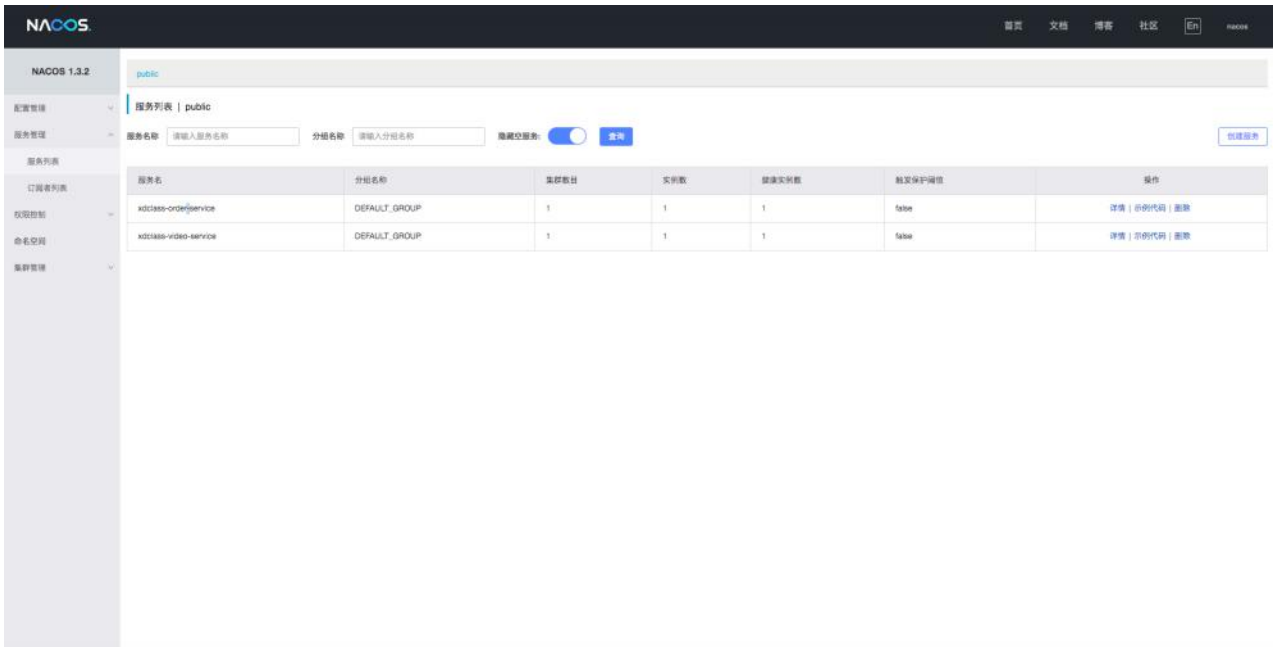

折叠
展开
100%
Nacos地图
特性大图
服务发现
服务发现
权重管理
打标管理
优雅上下线
配置管理
在线编辑
历史版本
一键回滚
灰度发布
推送轨迹
DNS服务
架构大图
逻辑架构图（介绍分层及组件构成）
领域模型（业务模型+数据结构）
部署架构图（从小到大演进图）
类视图（核心类关系）
业务大图
服务发现
配置管理
路由规则
限流规则
动态数据源
预案
开关
动态UI
战略大图
平台优势
运营优势
生态优势
优势大图
简单易用
经过几万人使用反馈优化，几代人打磨
特性丰富
十年软负载在这个领域积累丰富特性
超高性能
苛刻场景要求极致性能（AP）
超大容量
阿里规模经济造就强大容量
高可用
双十一不能失败，造就了高可用能力
生态大图
spring-cloud生态
dubbo生态
k8s生态
nodejs生态（egg/midway）

第2集 AlibabaCloud注册中心Nacos实战

简介：介绍什么是Nacos和搭建实战

- 注册中心Nacos介绍
- 官网：https://nacos.io/zh-cn/

- Linux/Mac安装Nacos
  - 解压安装包
  - 进入bin目录
  - 启动 sh startup.sh -m standalone
  - 访问 localhost:8848/nacos
  - 默认账号密码 nacos/nacos



第3集 基于Nacos实现订单-视频服务之间的调用

简介：项目集成Nacos实现服务直接的调用

- 视频服务集成Nacos
  - 添加依赖

```
<!--添加nacos客户端-->
<dependency>
      <groupId>com.alibaba.cloud</groupId>
      <artifactId>spring-cloud-starter-
alibaba-nacos-discovery</artifactId>
</dependency>
```

- 配置Nacos地址

```
server:
  port: 9000

spring:
  application:
    name: xdclass-video-service
  cloud:
    nacos:
      discovery:
        server-addr: 127.0.0.1:8848
```

- 启动类增加注解

```
@EnableDiscoveryClient
```

- 订单服务集成Nacos
- 用户服务集成Nacos
- 服务之间的调用

```
    @Autowired
    private DiscoveryClient discoveryClient;

    @Autowired
    private RestTemplate restTemplate;

    @RequestMapping("save")
```

```
    public VideoOrder save(int videoId){

        VideoOrder videoOrder = new
VideoOrder();
        videoOrder.setVideoId(videoId);

        List<ServiceInstance> list =
discoveryClient.getInstances("xdclass-video-
service");

        ServiceInstance serviceInstance =
list.get(0);

        Video video =
restTemplate.getForObject("http://"+serviceInst
ance.getHost()+":"+serviceInstance.getPort()+
                "/api/v1/video/find_by_id?
videoId="+videoId,Video.class);


videoOrder.setVideoTitle(video.getTitle());
        videoOrder.setVideoId(video.getId());
        return videoOrder;

    }
```

第4集 大话常见的负载均衡策略和解决方案

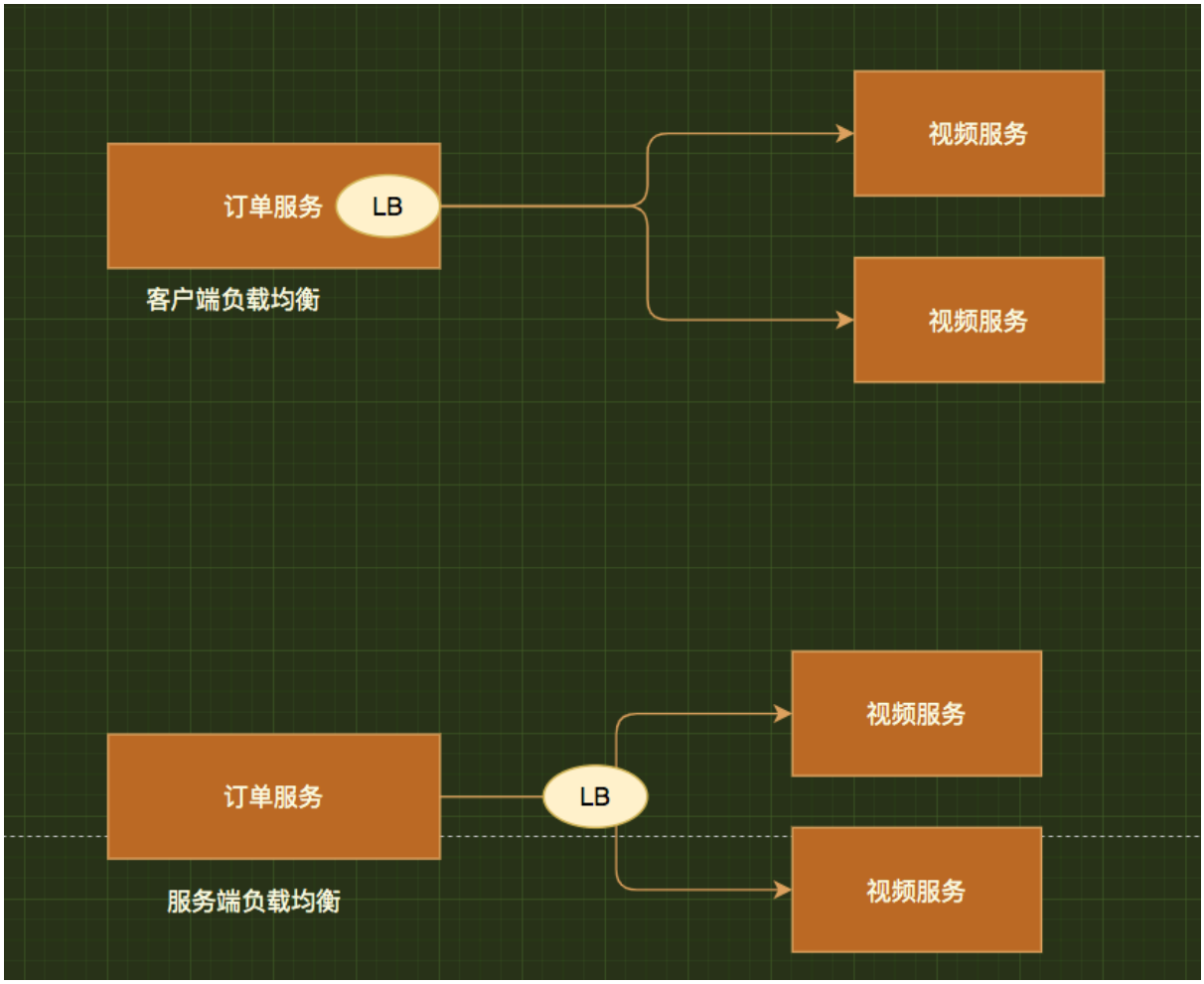
简介：讲解什么负载均衡和常见的解决方案

- 什么是负载均衡（Load Balance）

  > 分布式系统中一个非常重要的概念，当访问的服务具有多个实例时，需要根据某种“均衡”的策略决定请求发往哪个节点，这就是所谓的负载均衡，
  > 原理是将数据流量分摊到多个服务器执行，减轻每台服务器的压力，从而提高了数据的吞吐量

- 软硬件角度负载均衡的种类
  - 通过硬件来进行解决，常见的硬件有NetScaler、F5、Radware和Array等商用的负载均衡器，但比较昂贵的
  - 通过软件来进行解决，常见的软件有LVS、Nginx等,它们是基于Linux系统并且开源的负载均衡策略

- 从端的角度负载均衡有两种
  - 服务端负载均衡
  - 客户端负载均衡



- 常见的负载均衡策略（看组件的支持情况）
  - 节点轮询
    - 简介：每个请求按顺序分配到不同的后端服务器
  - weight 权重配置
    - 简介：weight和访问比率成正比，数字越大，分配得到的流量越高
  - 固定分发
    - 简介：根据请求按访问ip的hash结果分配，这样每个用户就可以固定访问一个后端服务器
  - 随机选择、最短响应时间等等

第5集 AlibabaCloud集成Ribbon实现负载均衡

简介：AlibabaCloud集成Ribbon实现负载均衡

- 什么是Ribbon Ribbon是一个客户端负载均衡工具，通过Spring Cloud封装，可以轻松和AlibabaCloud整合
- 订单服务增加@LoadBalanced 注解

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

- 调用实战

```
Video video =
restTemplate.getForObject("http://xdclass-
video-service/api/v1/video/find_by_id?
videoId="+videoId, Video.class);
```

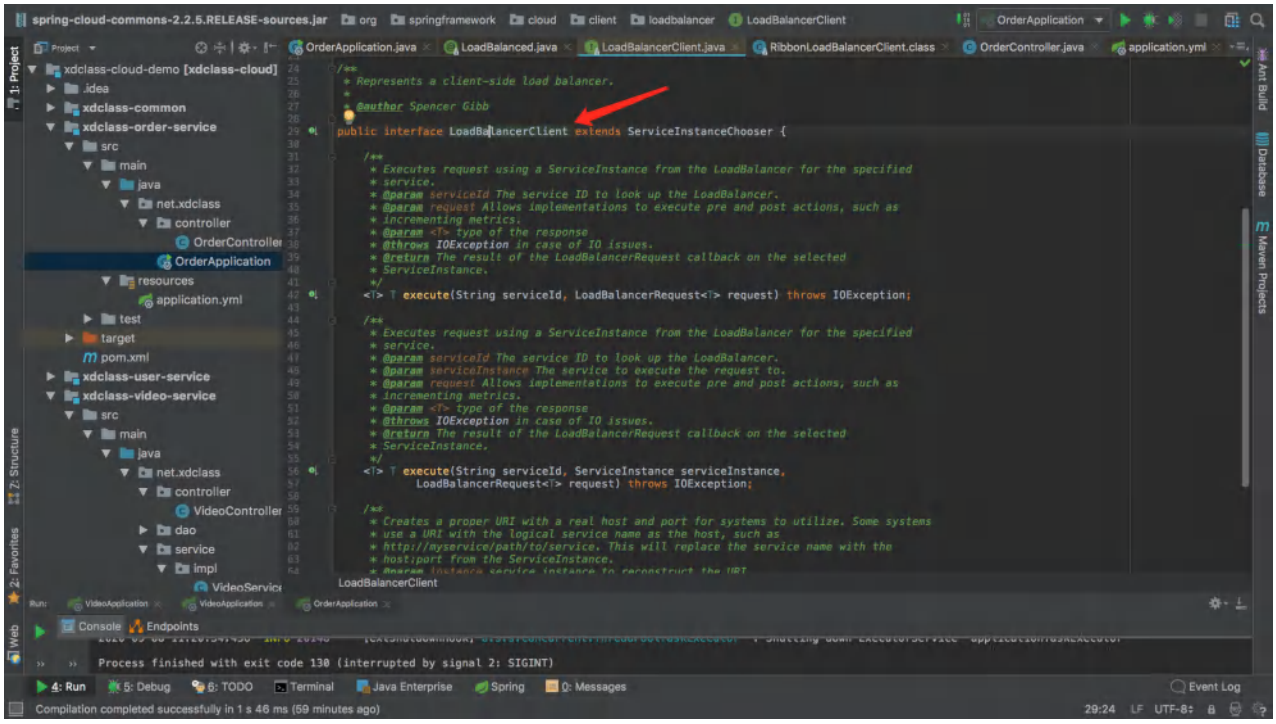
注意：方便大家看到负载均衡效果，在video类增加这个字段，记录当前机器ip+端口

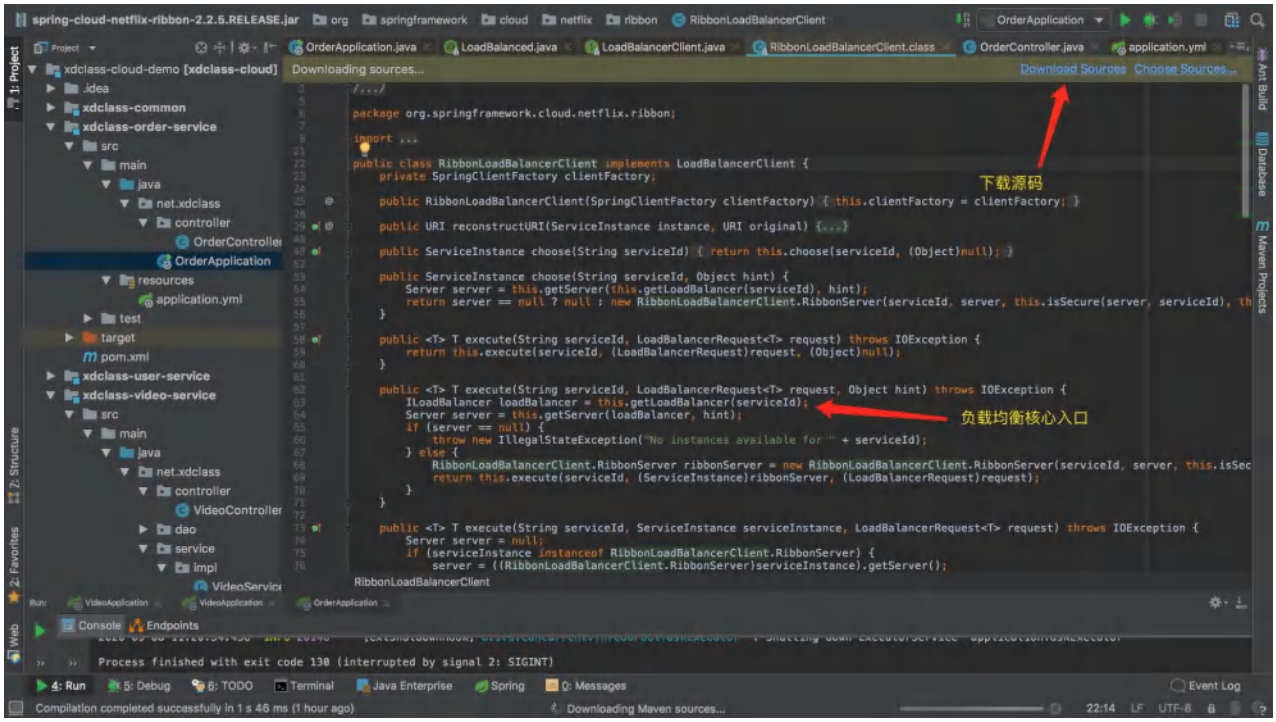
第五章 【高级篇幅】负载均衡进阶之Ribbon和Feign实战+源码分析

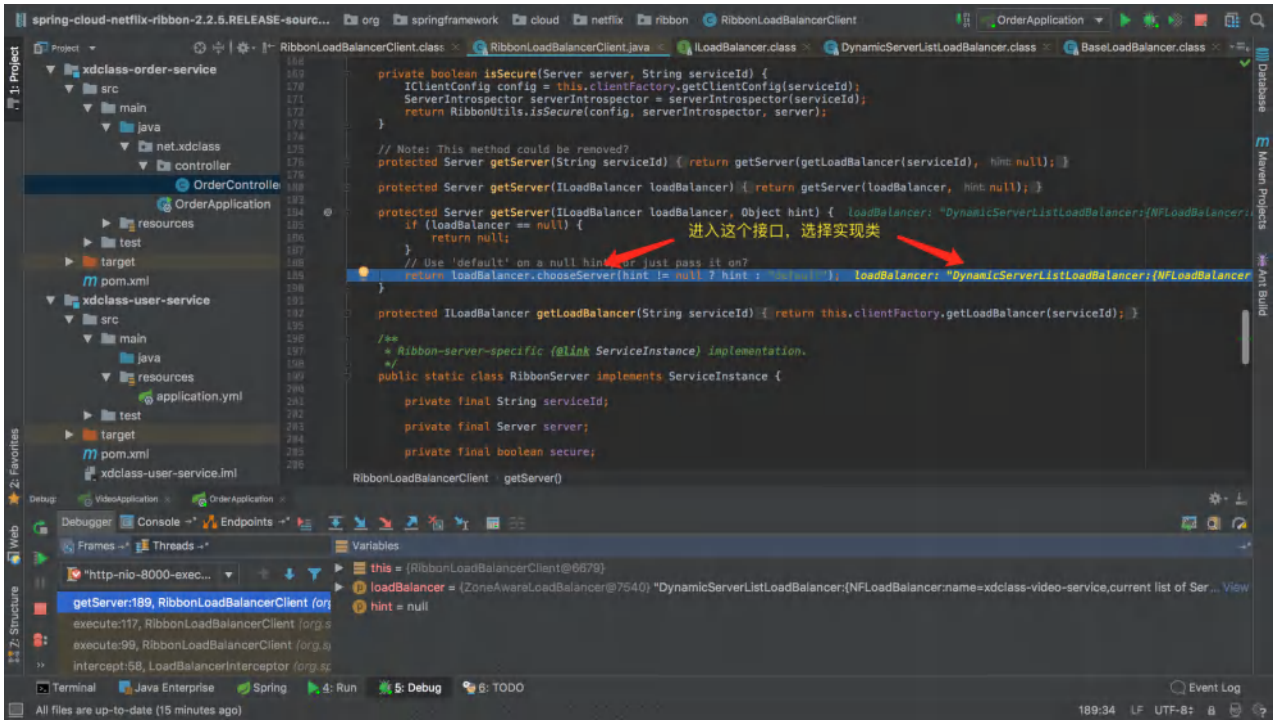
第1集 【源码剖析】手把手教学-Ribbon负载均衡源码实战

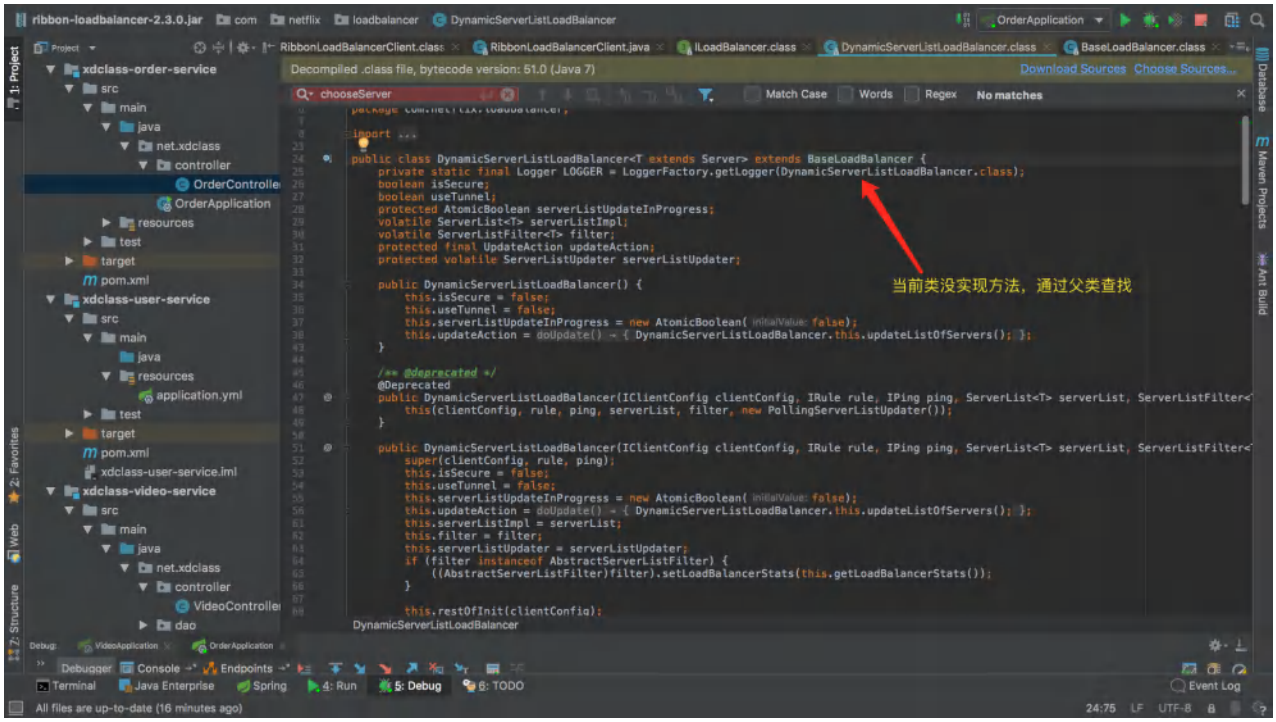
简介: 讲解ribbon服务间调用负载均衡源码分析

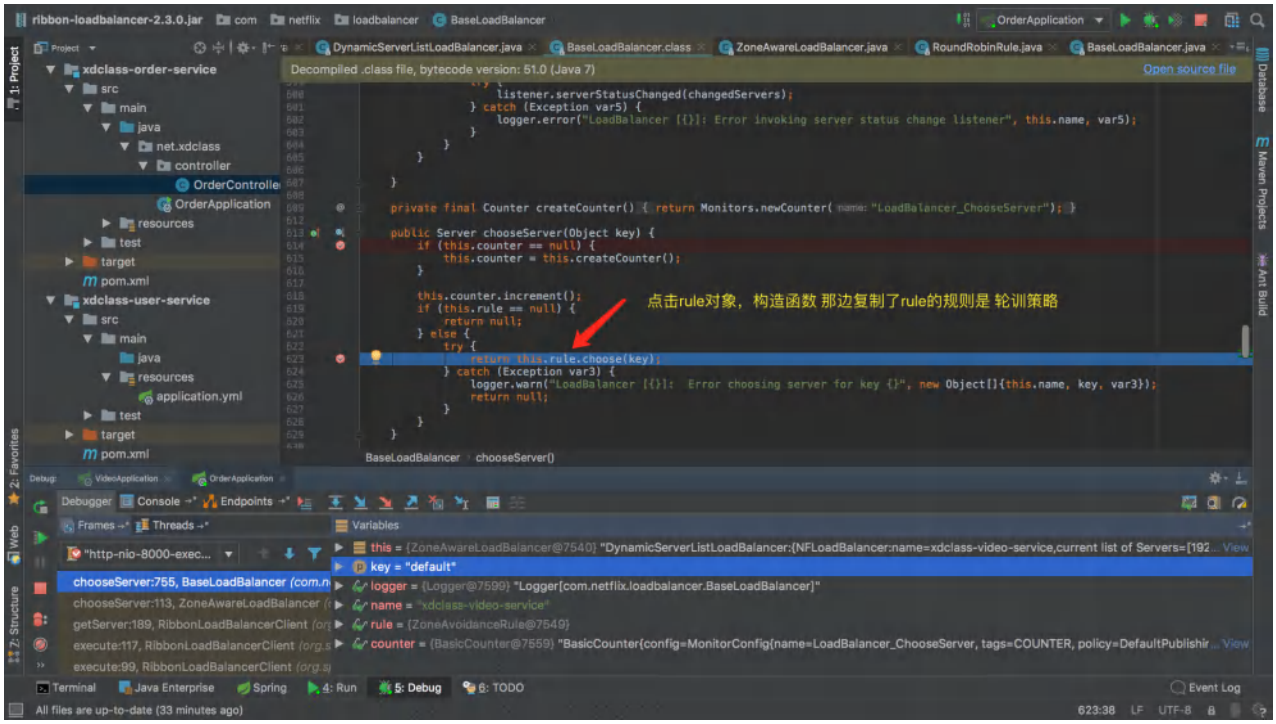
- 源码分析思路
  - 通过直接找入口
- 分析@LoadBalanced 1）首先从注册中心获取provider的列表 2）通过一定的策略选择其中一个节点 3）再返回给restTemplate调用




下载源码
负载均衡核心入口


进入这个接口，选择实现类

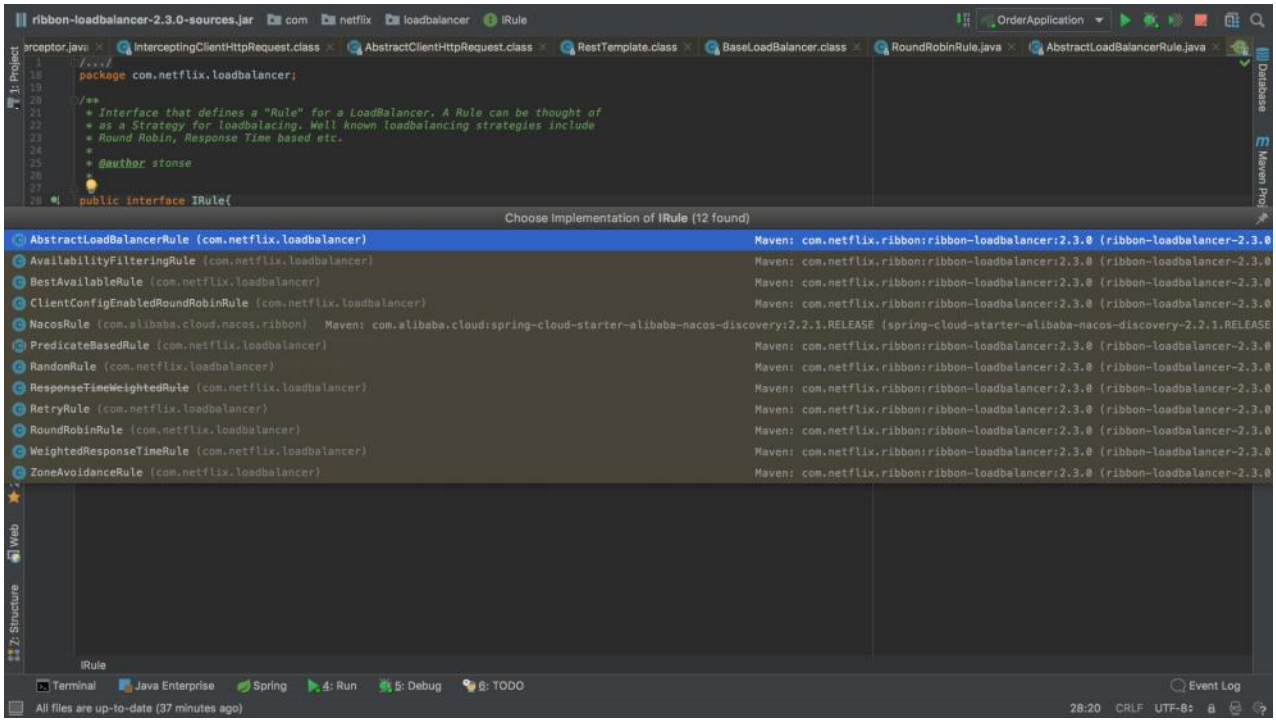

当前类没实现方法，通过父类查找


点击rule对象，构造函数 那边复制了rule的规则是 轮训策略

第2集 高级篇之AlibabaCloud负载均衡策略调整实战

简介:自定义Ribbon负载均衡策略实战

- 源码分析知道ribbon支持多种负载均衡策略



- Ribbon支持的负载均衡策略介绍

策略类	命名	描述
RandomRule	随机策略	随机选择server
RoundRobinRule	轮询策略	按照顺序选择server（默认）
RetryRule	重试策略	当选择server不成功，短期内尝试选择一个可用的server

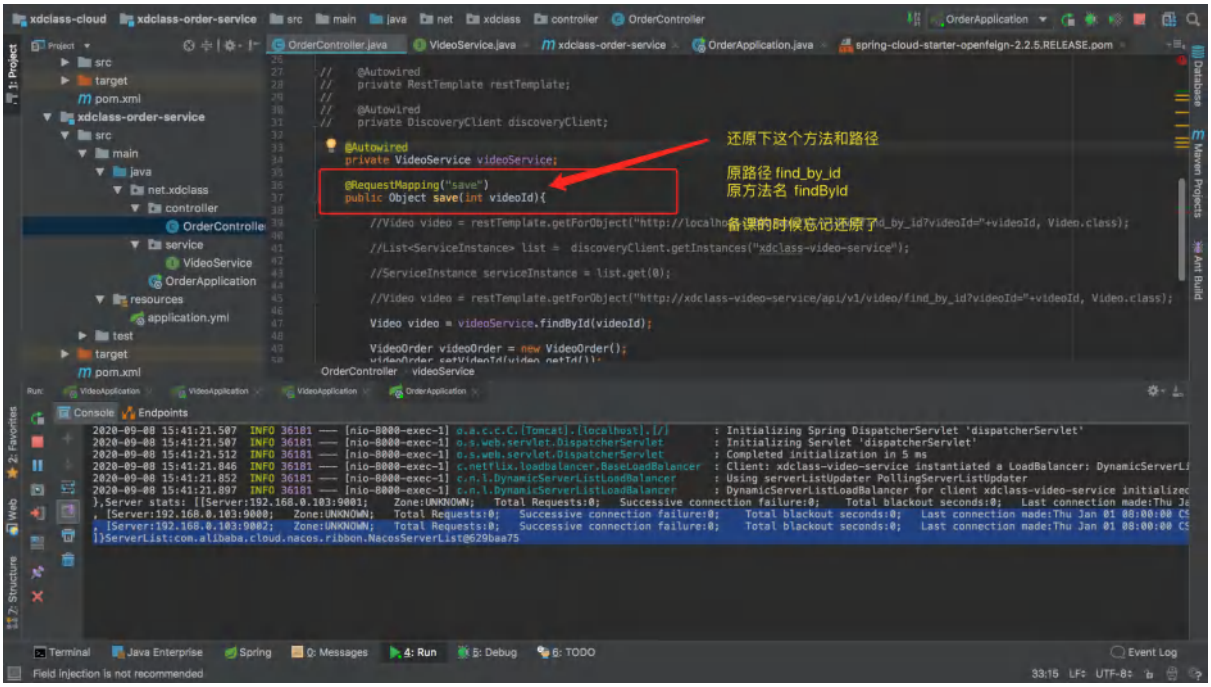
AvailabilityFilteringRule	可用过滤策略	过滤掉一直失败并被标记为circuit tripped的server，过滤掉那些高并发链接的server（active connections超过配置的阈值）
WeightedResponseTimeRule	响应时间加权重策略	根据server的响应时间分配权重，以响应时间作为权重，响应时间越短的服务器被选中的概率越大，综合了各种因素，比如：网络，磁盘，io等，都直接影响响应时间
ZoneAvoidanceRule	区域权重策略	综合判断server所在区域的性能，和server的可用性，轮询选择server

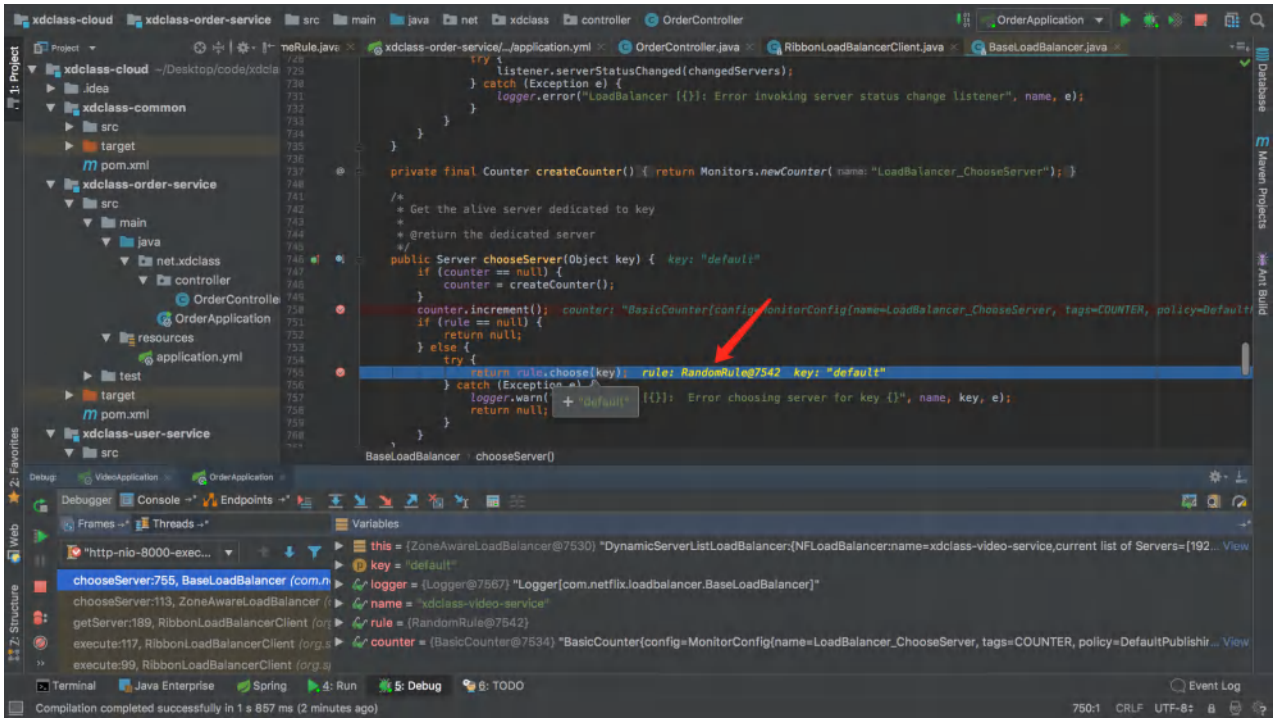
- 负载均衡策略调整实战

订单服务增加配置

```
xdclass-video-service:
  ribbon:
    NFLoadBalancerRuleClassName:
com.netflix.loadbalancer.RandomRule
```

- 注意：订单服务中api方法名那边个save方法，还原为findById，备课的时候忘记还原，不影响使用





- 策略选择： 1、如果每个机器配置一样，则建议不修改策略 (推荐) 2、如果部分机器配置强，则可以改为 WeightedResponseTimeRule

第3集 微服务新一代负载均衡组件Open-Feign介绍

简介：讲解新一代负载均衡组件feign介绍

- 原先ribbon代码存在的问题：不规范，风格不统一，维护性比较差
- 什么是Feign:

  > SpringCloud提供的伪http客户端(本质还是用http)，封装了Http调用流程，更适合面向接口化
  > 让用Java接口注解的方式调用Http请求.
  >
  > 不用像Ribbon中通过封装HTTP请求报文的方式调用 Feign默认集成了Ribbon

- 官方文档
  - https://spring.io/projects/spring-cloud-openfeign
  - 版本 2.2.5
- Nacos支持Feign,可以直接集成实现负载均衡的效果

第4集 改造微服务 集成Open-Feign实现远程方法调用

简介：改造微服务 集成Feign实现远程方法调用

- Feign让方法调用更加解耦
- 使用feign步骤讲解
    - 加入依赖

```
<dependency>

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

- 配置注解

```
启动类增加@EnableFeignClients
```

- 增加一个接口

```
订单服务增加接口，服务名称记得和nacos保持一样
@FeignClient(name="xdclass-video-service")
```

- 编写代码

```
@GetMapping(value = "/api/v1/video/find_by_id")
Video findById(@RequestParam("videoId") int videoId);
```

第5集 post方式对象传输 Open-Feign 实现远程方法调用

简介：改造微服务 集成Feign实现远程方法调用

- GET方式查询简单
- POST方式提交怎么做

```
//订单服务这边
@PostMapping(value = "/api/v1/video/save")
Video saveVideo(@RequestBody Video video);


@PostMapping("save")
public Object save(@RequestBody Video video){
        System.out.println(video.getTitle());
        return  video;
}
```

    - 注意：
        - 路径 核对
        - Http方法必须对应
        - 使用服务提供方用@RequestBody，要使用@PostMapping

- Ribbon和feign两个的区别和选择

```
选择feign
默认集成了ribbon
写起来更加思路清晰和方便
采用注解方式进行配置，配置熔断等方式方便
```

旭瑶&小滴课堂 愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

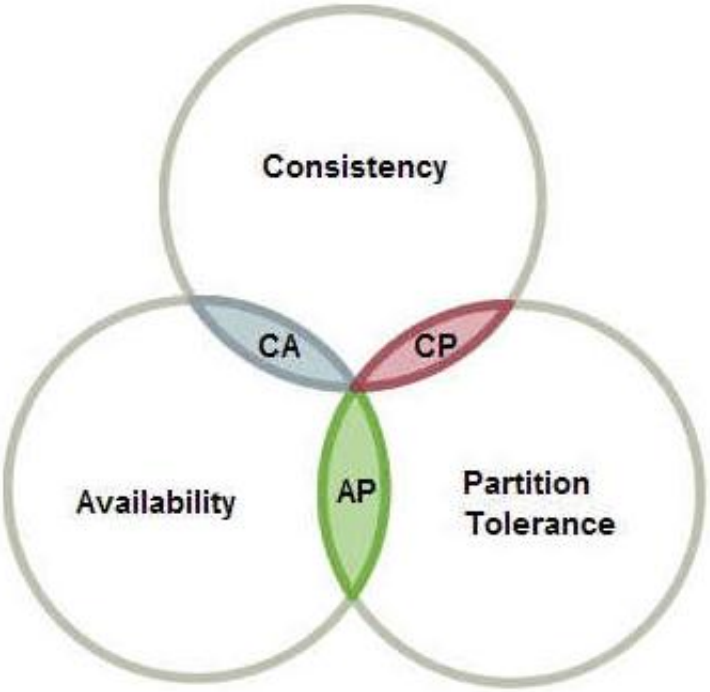
第六章 【面试题】想成为架构师的你不可不知道的分布式架构理论

第1集 【面试题】掌握微服务你必须知道的CAP理论

简介：讲解分布式应用核心CAP知识

- 可能会有疑惑，可以看多几遍
- CAP定理: 指的是在一个分布式系统中，Consistency（一致性）、 Availability（可用性）、Partition tolerance（分区容错性），三者不可同时获得
  - 一致性（C）： 所有节点都可以访问到最新的数据
  - 可用性（A）： 每个请求都是可以得到响应的，不管请求是成功还是失败
  - 分区容错性（P）： 除了全部整体网络故障，其他故障都不能导致整个系统不可用

- CAP理论就是说在分布式存储系统中，最多只能实现上面的两点。而由于当前的网络硬件肯定会出现延迟丢包等问题，所以分区容忍性是我们必须需要实现的。所以我们只能在一致性和可用性之间进行权衡



CA： 如果不要求P（不允许分区），则C（强一致性）和A（可用性）是可以保证的。但放弃P的同时也就意味着放弃了系统的扩展性，也就是分布式节点受限，没办法部署子节点，这是违背分布式系统设计的初衷的

CP：如果不要求A（可用），每个请求都需要在服务器之间保持强一致，而P（分区）会导致同步时间无限延长(也就是等待数据同步完才能正常访问服务)，一旦发生网络故障或者消息丢失等情况，就要牺牲用户的体验，等待所有数据全部一致了之后再让用户访问系统

AP：要高可用并允许分区，则需放弃一致性。一旦分区发生，节点之间可能会失去联系，为了高可用，每个节点只能用本地数据提供服务，而这样会导致全局数据的不一致性。

第2集 【面试题】CAP里面下的注册中心选择思考

简介：讲解常见的分布式核心CAP理论介绍

- 常见注册中心：zk、eureka、nacos
- 那你应该怎么选择

	Nacos	Eureka	Consul	Zookeeper
一致性协议	CP+AP	AP	CP	CP
健康检查	TCP/HTTP/MYSQL/Client Beat	心跳	TCP/HTTP/gRPC/Cmd	Keep Alive
雪崩保护	有	有	无	无
访问协议	HTTP/DNS	HTTP	HTTP/DNS	TCP
SpringCloud集成	支持	支持	支持	支持

- Zookeeper：CP设计，保证了一致性，集群搭建的时候，某个节点失效，则会进行选举行的leader，或者半数以上节点不可用，则无法提供服务，因此可用性没法满足
- Eureka：AP原则，无主从节点，一个节点挂了，自动切换

其他节点可以使用，去中心化

- 结论：
  - 分布式系统中P,肯定要满足，所以只能在CA中二选一
  - 没有最好的选择，最好的选择是根据业务场景来进行架构设计
  - 如果要求一致性，则选择zookeeper/Nacos，如金融行业 CP
  - 如果要求可用性，则Eureka/Nacos，如电商系统 AP
  - CP： 适合支付、交易类，要求数据强一致性，宁可业务不可用，也不能出现脏数据
  - AP: 互联网业务，比如信息流架构，不要求数据强一致，更想要服务可用

第3集 【面试题】一致性和可用性的权衡结果 BASE理论

简介：讲解分布式CAP的权衡结果 BASE理论

- 什么是Base理论

> CAP 中的一致性和可用性进行一个权衡的结果，核心思想就是：我们无法做到强一致，但每个应用都可以根据自身的业务特点，采用适当的方式来使系统达到最终一致性，来自 ebay 的架构师提出

- Basically Available(基本可用)
  - 假设系统，出现了不可预知的故障，但还是能用, 可能会有性能或者功能上的影响

- Soft state（软状态）
  - 允许系统中的数据存在中间状态，并认为该状态不影响系统的整体可用性，即允许系统在多个不同节点的数据副本存在数据延时

- Eventually consistent（最终一致性）
  - 系统能够保证在没有其他新的更新操作的情况下，数据最终一定能够达到一致的状态，因此所有客户端对系统的数据访问最终都能够获取到最新的值

第七章 高并发下的微服务架构存在的问题和解决方案

第1集 海量请求下的微服务架构存在的问题

简介：高并发下的微服务存在的问题

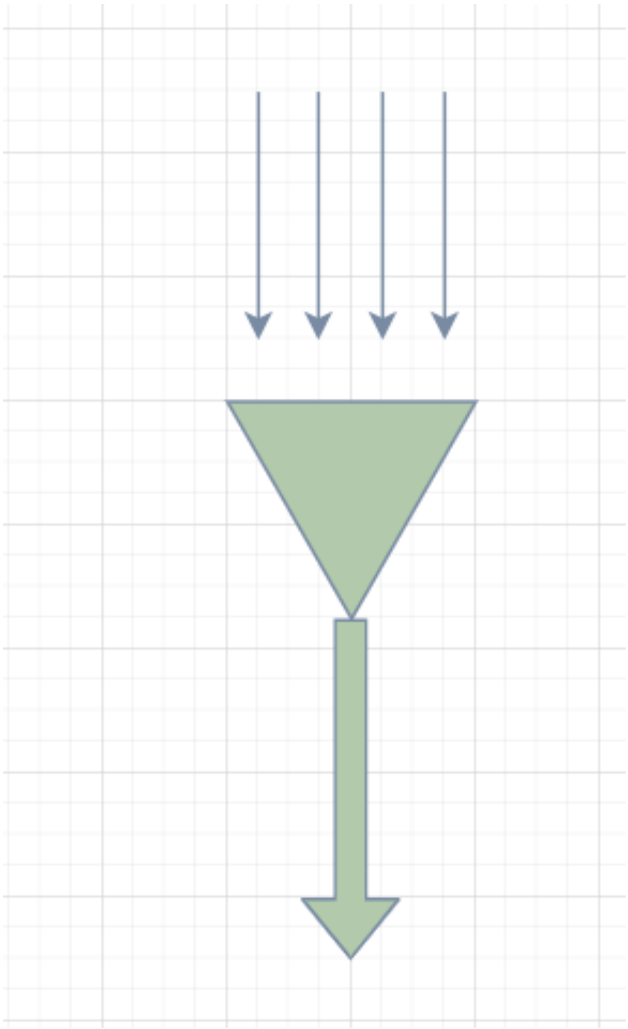
- 高并发下存在的问题
  - 微服务拆分多个系统，服务之间互相依赖，可能会由于系统负载过高，突发流量或者网络等各种异常情况 导致服务不可用。

- 核心思想-面向失败编程
  - 不要外界影响
  - 不被请求拖垮
    - 上游服务
    - 下游服务

第2集 面向失败编程-微服务架构容错方案介绍

简介：高并发下的微服务容错方案

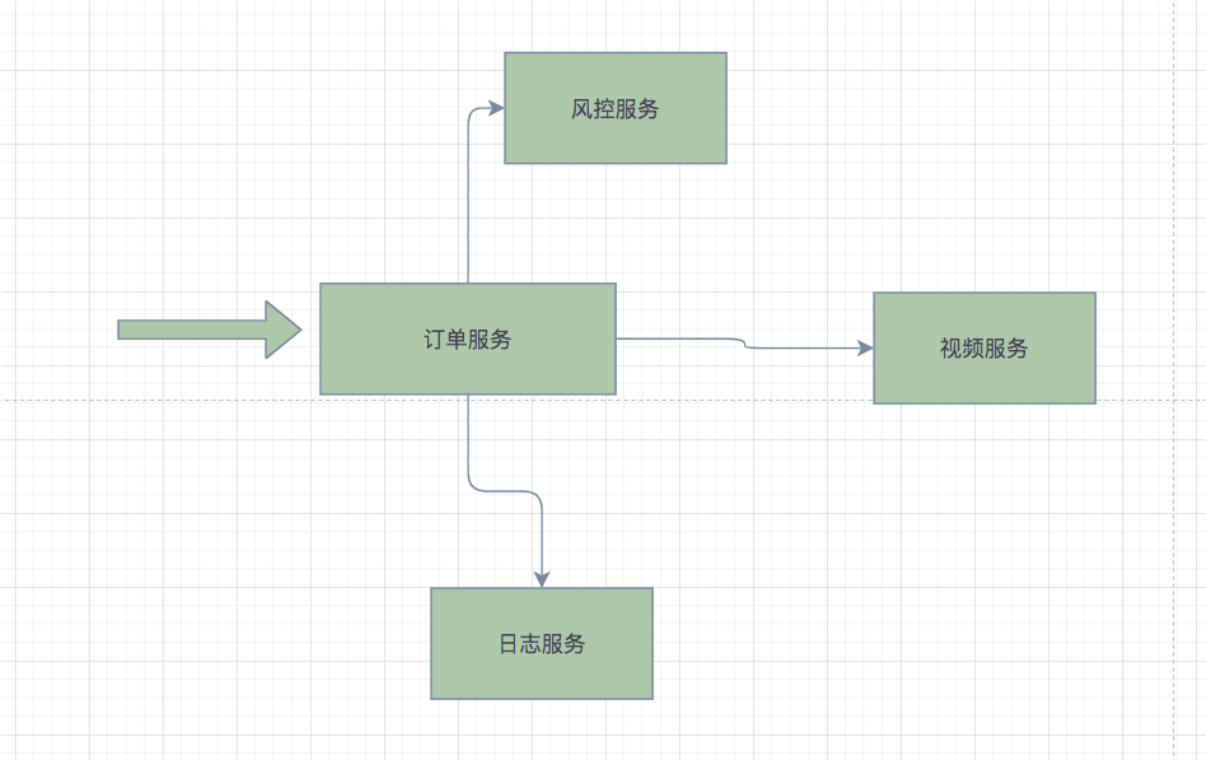
- 限流
- 漏斗，不管流量多大，均匀的流入容器，令牌桶算法，漏桶算法



- 熔断：
  - 保险丝，熔断服务，为了防止整个系统故障，包含当前和下游服务 下单服务 -》商品服务-》用户服务 -》（出现异常-》熔断风控服务
- 降级：
  - 抛弃一些非核心的接口和数据，返回兜底数据 旅行箱的例子：只带核心的物品，抛弃非核心的，等有条件的时候再去携带这些物品
- 隔离：
  - 服务和资源互相隔离，比如网络资源，机器资源，线程资源等，不会因为某个服务的资源不足而抢占其他服务

的资源

- 熔断和降级互相交集
  - 相同点：
    - 从可用性和可靠性触发，为了防止系统崩溃
    - 最终让用户体验到的是某些功能暂时不能用
  - 不同点
    - 服务熔断一般是下游服务故障导致的，而服务降级一般是从整体系统负荷考虑，由调用方控制
- 想进行微服务的容错，业界目前有Sentinel、Hystrix，相对于AlibabaCloud而言，Sentinel是最好的搭配



第3集 新版分布式系统的流量防卫兵-Sentinel介绍

简介：介绍什么是分布式系统的流量防卫兵Sentinel

- 什么是Sentinel
  - 阿里巴巴开源的分布式系统流控工具
  - 以流量为切入点，从流量控制、熔断降级、系统负载保护等多个维度保护服务的稳定性
  - 丰富的应用场景：消息削峰填谷、集群流量控制、实时熔断下游不可用应用等
  - 完备的实时监控：Sentinel 同时提供实时的监控功能
  - 提供开箱即用的与其它开源框架/库的整合模块，例如与 Spring Cloud、Dubbo、gRPC 的整合
- 官网：https://github.com/alibaba/Sentinel/wiki/%E4%BB%8B%E7%BB%8D
- Sentinel版本：2.2.1

- 核心概念：
  - 资源：是 Sentinel 中的核心概念之一，可以是java程序中任何内容，可以是服务或者方法甚至代码，总结起来就是我们要保护的东西
  - 规则：定义怎样的方式保护资源，主要包括流控规则、熔断降级规则等

第4集 流量防卫兵-Sentinel依赖引入和控制台搭建

简介：微服务引入Sentinel和控制台搭建

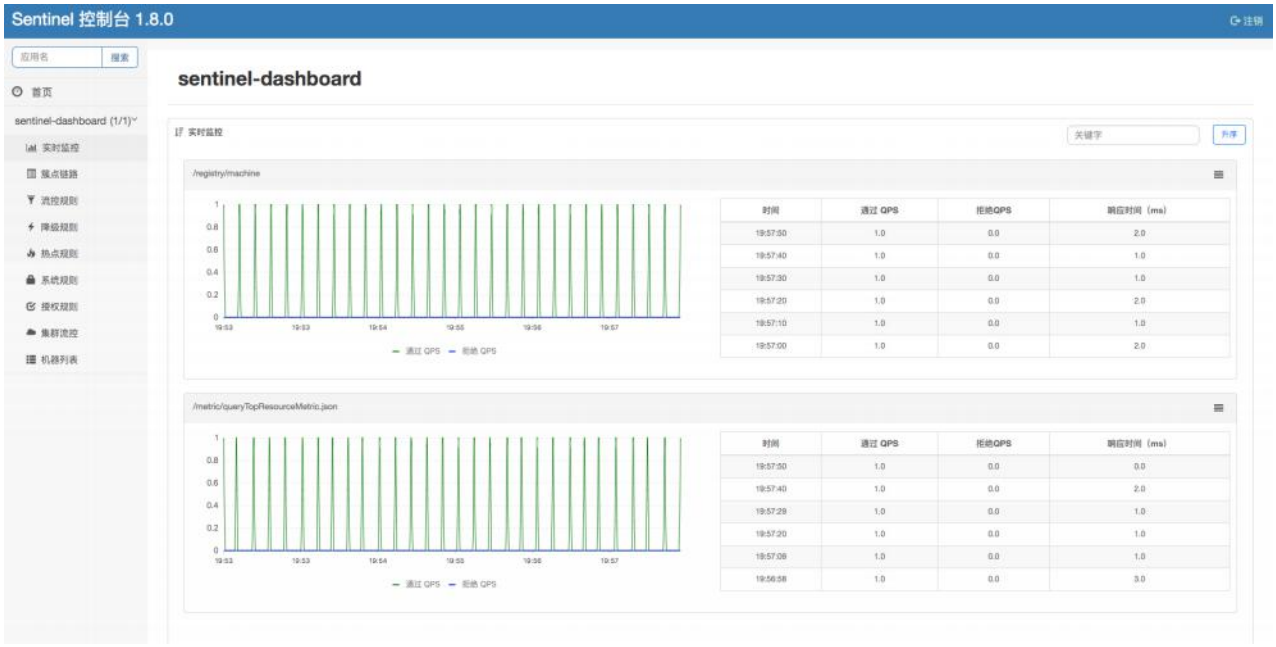
- Sentinel 分为两个部分
  - 核心库（Java 客户端）不依赖任何框架/库，能够运行于所有 Java 运行时环境，同时对 Dubbo、Spring Cloud 等框架也有较好的支持。
  - 控制台（Dashboard）基于 Spring Boot 开发，打包后可以直接运行，不需要额外的 Tomcat 等应用容器。
- 微服务引入Sentinel依赖

```
 <dependency>

<groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-
alibaba-sentinel</artifactId>
</dependency>
```

- Sentinel控制台搭建
- 文档：https://github.com/alibaba/Sentinel/wiki/控制台
- 控制台包含如下功能:
  - 查看机器列表以及健康情况：收集 Sentinel 客户端发送的心跳包，用于判断机器是否在线。
  - 监控 (单机和集群聚合)通过 Sentinel 客户端暴露的监控 API，定期拉取并且聚合应用监控信息，最终可以实现秒级的实时监控。
  - 规则管理和推送：统一管理推送规则。
  - 鉴权：生产环境中鉴权非常重要。这里每个开发者需要根据自己的实际情况进行定制。

> 注意：Sentinel 控制台目前仅支持单机部署

```
//启动 Sentinel 控制台需要 JDK 版本为 1.8 及以上版本,
//-Dserver.port=8080 用于指定 Sentinel 控制台端口为 8080
//默认用户名和密码都是 sentinel

java -Dserver.port=8080 -Dcsp.sentinel.dashboard.server=localhost:8080 -Dproject.name=sentinel-dashboard -jar sentinel-dashboard-1.8.0.jar
```



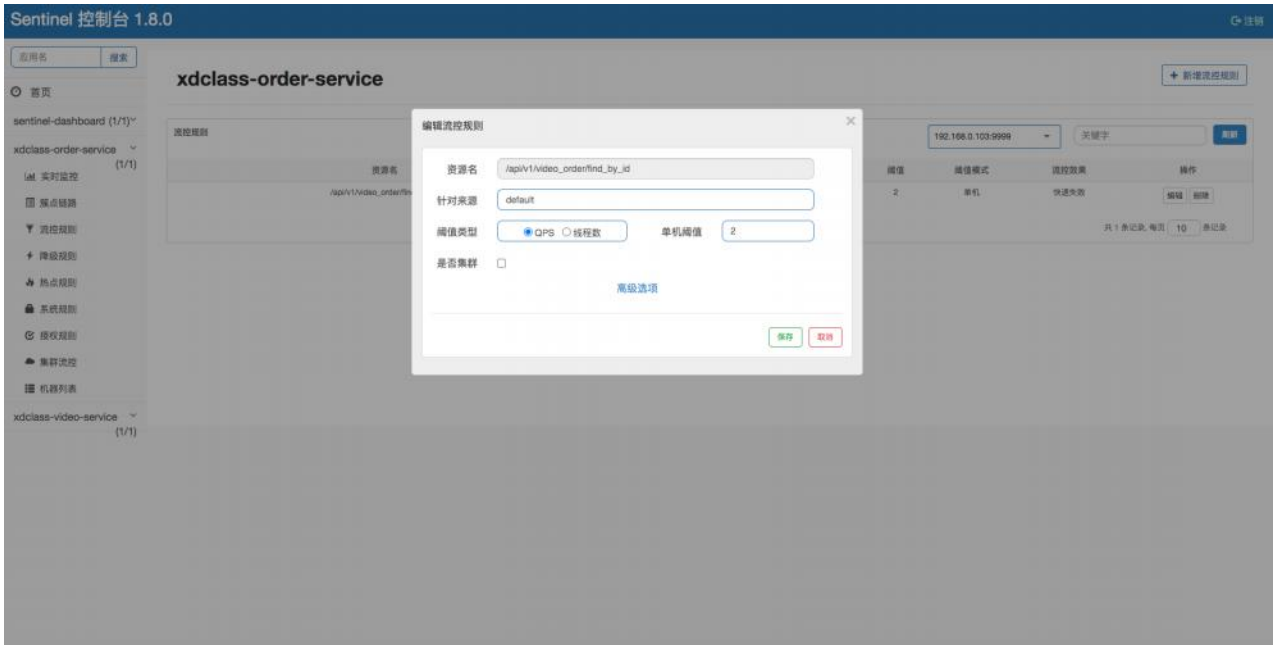
第5集 AliababCloud微服务整合Sentinel限流配置实战

简介：讲解AliababCloud微服务整合Sentinel限流配置实操

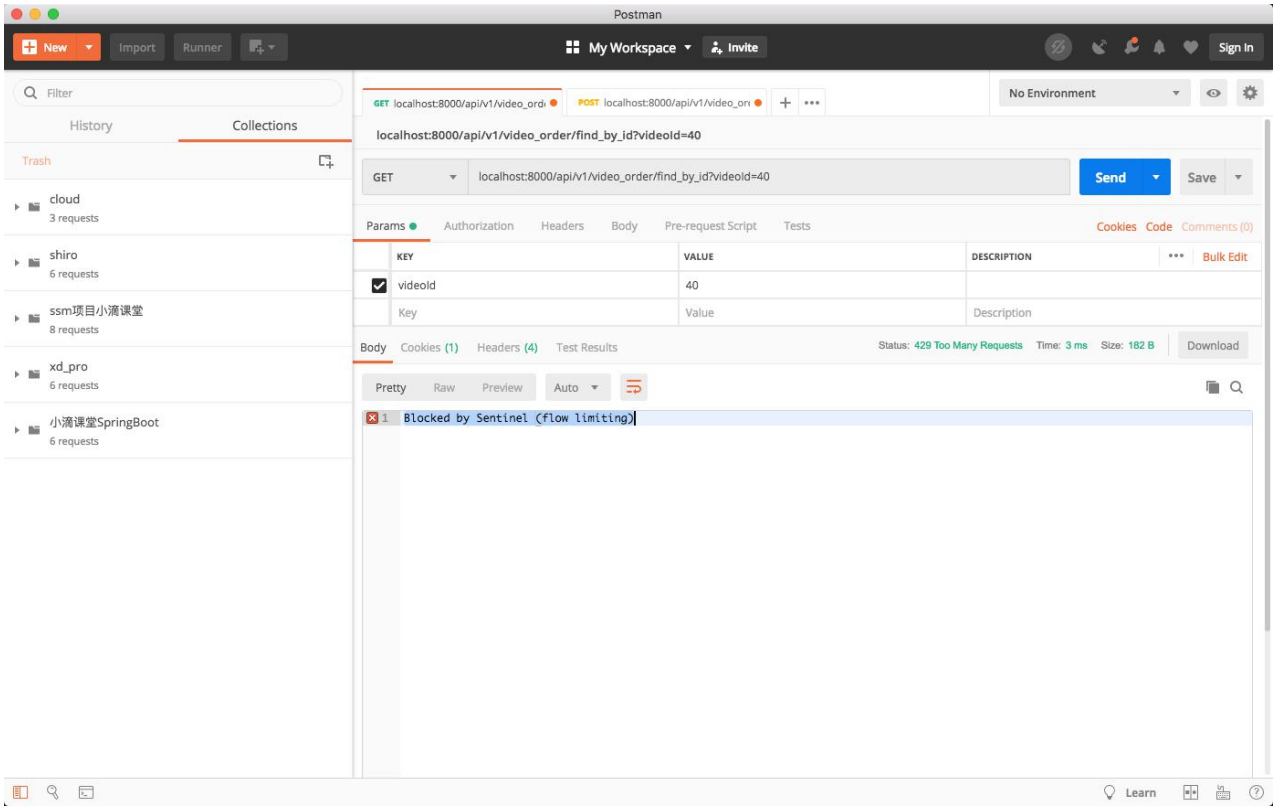
- 多个微服务接入Sentinel配置

```
spring:
  cloud:
    sentinel:
      transport:
        dashboard: 127.0.0.1:8080
        port: 9999

#dashboard: 8080 控制台端口
#port: 9999 本地启的端口，随机选个不能被占用的，与dashboard进行数据交互，会在应用对应的机器上启动一个Http Server，该 Server 会与 Sentinel 控制台做交互，若被占用,则开始+1一次扫描
```

- 微服务注册上去后，由于Sentinel是懒加载模式，所以需要访问微服务后才会在控制台出现
- 限流配置实操
  - 控制台配置


Sentinel 控制台 1.8.0
xdclass-order-service
编辑流控规则
资源名
/api/v1/video_order/find_by_id
针对来源
default
阈值类型
QPS
线程数
单机阈值
2
是否集群
高级选项
保存
取消

- 浏览器刷新


Postman
New
Import
Runner
My Workspace
Invite
Sign In
Filter
History
Collections
Trash
cloud
3 requests
shiro
6 requests
ssm项目小滴课堂
8 requests
xd_pro
6 requests
小滴课堂SpringBoot
6 requests
No Environment
localhost:8000/api/v1/video_order/find_by_id?videoId=40
GET
localhost:8000/api/v1/video_order/find_by_id?videoId=40
Send
Save
Params
Authorization
Headers
Body
Pre-request Script
Tests
Cookies
Code
KEY
VALUE
DESCRIPTION
Bulk Edit
videoId
40
Key
Value
Description
Body
Cookies (1)
Headers (4)
Test Results
Status: 429 Too Many Requests
Time: 3 ms
Size: 182 B
Download
Pretty
Raw
Preview
Auto
1 Blocked by Sentinel (flow limiting)
Learn

旭瑶&小滴课堂 愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第八章 【进阶篇幅】玩转Sentinel多种流空规则和实战

第1集 玩转Sentinel流量控制功能

简介：讲Sentinel流量控制详细操作

- 流量控制（flow control）
  - 原理是监控应用流量的 QPS 或并发线程数等指标，当达到指定的阈值时对流量进行控制，以避免被瞬时的流量高峰冲垮，从而保障应用的高可用性。

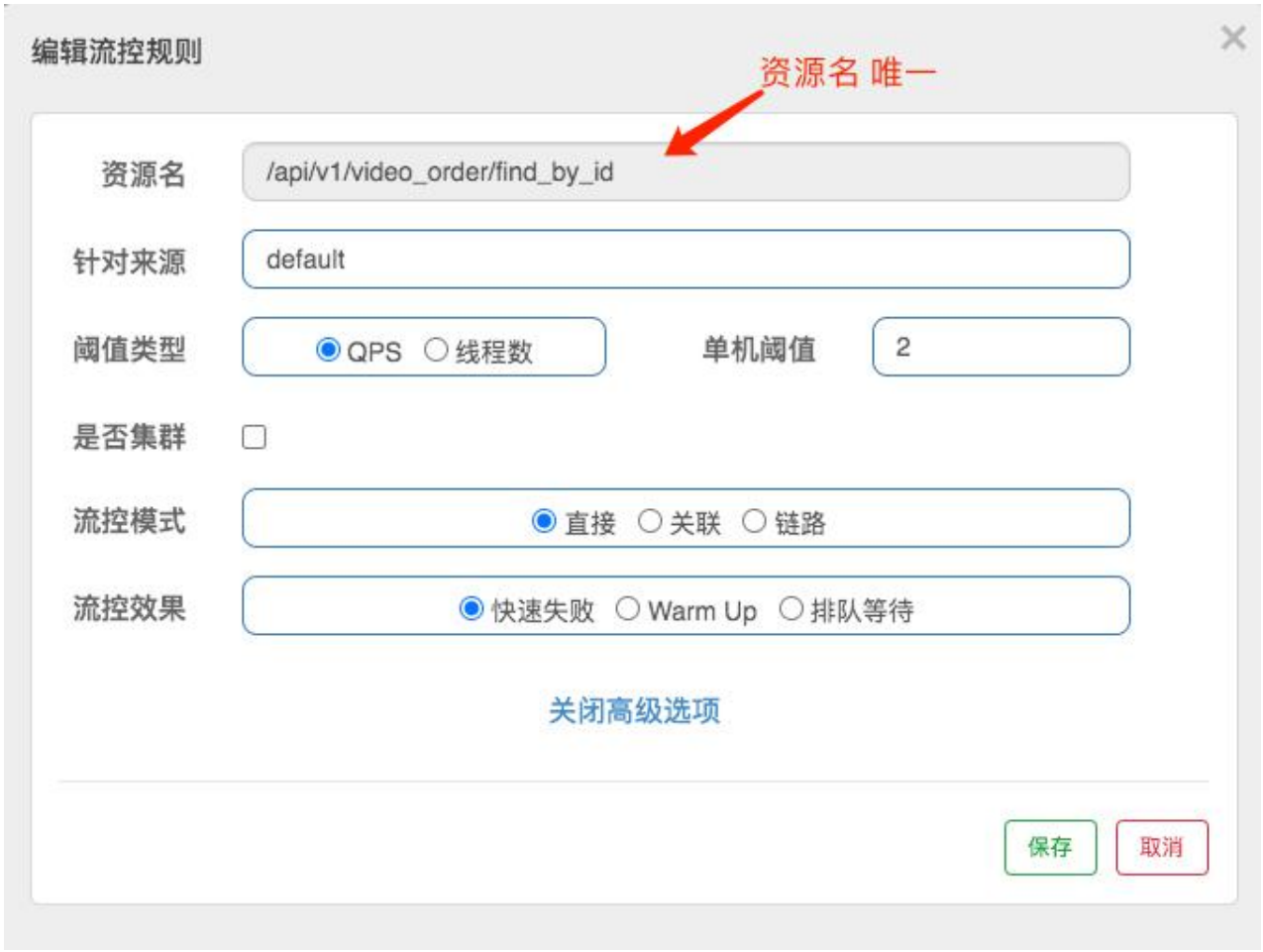
- 两种规则
  - 基于统计并发线程数的流量控制

> 并发数控制用于保护业务线程池不被慢调用耗尽
>
> `Sentinel` 并发控制不负责创建和管理线程池，而是简单统计当前请求上下文的线程数目（正在执行的调用数目）
>
> 如果超出阈值，新的请求会被立即拒绝，效果类似于信号量隔离。

  - 基于统计QPS的流量控制

> 当 `QPS` 超过某个阈值的时候，则采取措施进行流量控制

- 控制面板介绍
  - 资源名：默认是请求路径，可自定义
  - 针对来源：对哪个微服务进行限流，默认是不区分来源，全部限流，这个是针对 区分上游服务进行限流, 比如 视频服务 被 订单服务、用户服务调用，就可以针对来源进行限流



第2集 基于并发线程数进行限流配置实操

简介：基于并发线程进行限流配置实操

- 开发临时接口，方便测试

```
    @RequestMapping("list")
    public Object list(){
        try {
            TimeUnit.SECONDS.sleep(3);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Map<String,String> map  = new HashMap<>
();

        map.put("title1","ALibabaCloud微服务专
题");
        map.put("title2","小滴课堂面试专题第一季");

        return map;
    }
```

- 基于统计并发线程数的流量控制

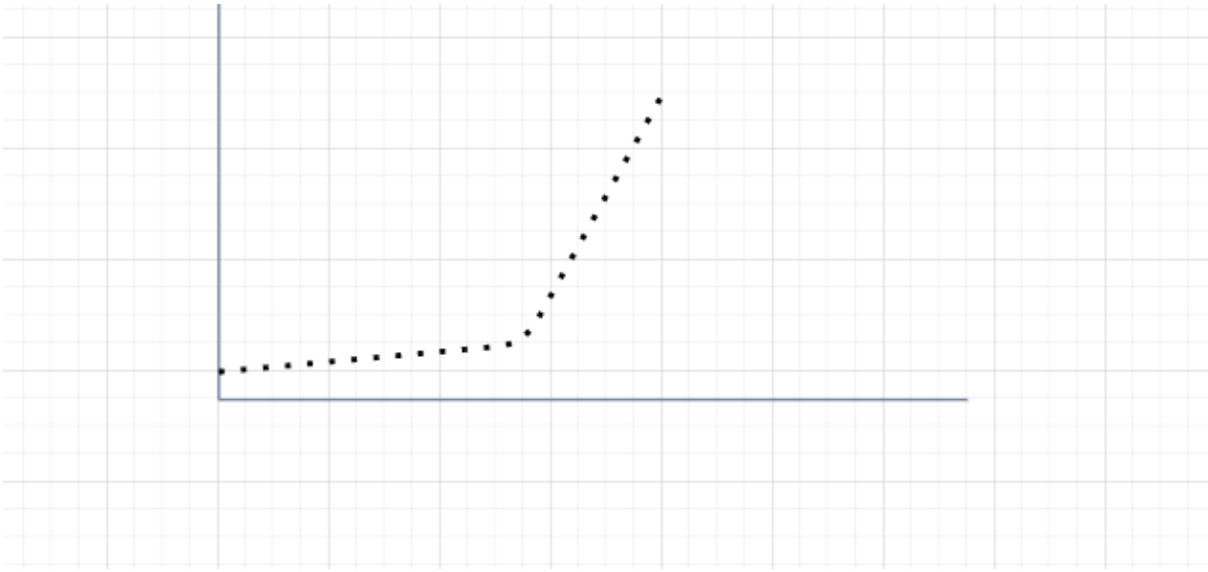
> 并发数控制用于保护业务线程池不被慢调用耗尽
>
> `Sentinel` 并发控制不负责创建和管理线程池，而是简单统计当前请求上下文的线程数目（正在执行的调用数目）
>
> 如果超出阈值，新的请求会被立即拒绝，效果类似于信号量隔离。并发数控制通常在调用端进行配置

- 流控规则会下发到微服务，微服务如果重启，则流控规则会消失可以持久化配置
- 选择阈值类型 ”线程数“ ，配置是1
- 刷新浏览器

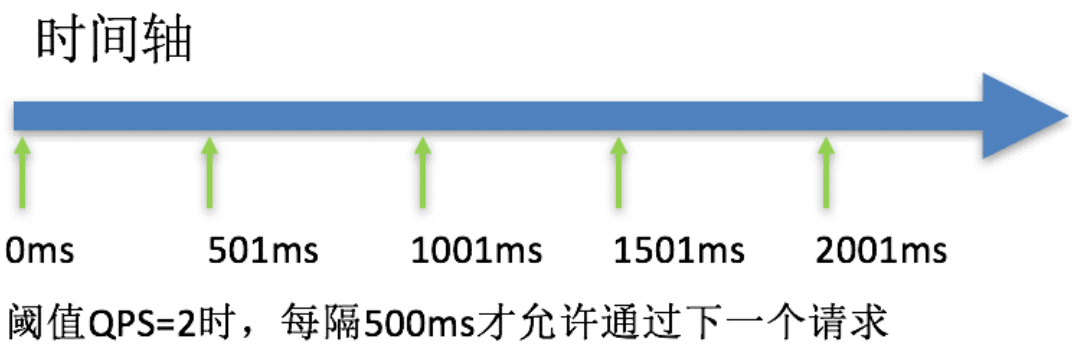
第3集 流控规则效果-直接拒绝-冷启动预热-匀速排队讲解

简介：基于并发线程进行限流配置实操

- 流量控制的效果包括以下几种：
  - 直接拒绝：默认的流量控制方式，当QPS超过任意规则的阈值后，新的请求就会被立即拒绝
  - Warm Up：冷启动/预热，如果系统在此之前长期处于空闲的状态，我们希望处理请求的数量是缓步的增多，经过预期的时间以后，到达系统处理请求个数的最大值



  - 匀速排队：严格控制请求通过的间隔时间，也即是让请求以均匀的速度通过，对应的是漏桶算法，主要用于处理间隔性突发的流量，如消息队列，想象一下这样的场景，在某一秒有大量的请求到来，而接下来的几秒则处于空闲状态，我们希望系统能够在接下来的空闲期间逐渐处理这些请求，而不是在第一秒直接拒绝多余的请求



    - 注意：
        - 匀速排队等待策略是 Leaky Bucket 算法结合虚拟队列等待机制实现的。
        - 匀速排队模式暂时不支持 QPS > 1000 的场景
- 流控文档
    - https://github.com/alibaba/Sentinel/wiki/流量控制#基于调用关系的限流

第4集 新版Sentinel-微服务高可用利器-熔断降级规则讲解

简介：讲解 微服务高可用利器Sentinel熔断降级规则

- 备注：如果 簇点链路 没数据，刷多几次接口
- 熔断降级（虽然是两个概念，基本都是互相配合）
  - 对调用链路中不稳定的资源进行熔断降级也是保障高可用的重要措施之一
  - 对不稳定的**弱依赖服务调用**进行熔断降级，暂时切断不稳定调用，避免局部不稳定因素导致整体的雪崩
  - 熔断降级作为保护自身的手段，通常在客户端（调用端）进行配置

- 什么是Sentinel降级规则
  - 文档：https://github.com/alibaba/Sentinel/wiki/熔断降级
  - 就是配置一定规则，然后满足之后就对服务进行熔断降级

- Sentinel 熔断策略

- 慢调用比例(响应时间): 选择以慢调用比例作为阈值，需要设置允许的慢调用 RT（即最大的响应时间），请求的响应时间大于该值则统计为慢调用
  - 比例阈值：修改后不生效-目前已经反馈给官方那边的bug
  - 熔断时长：超过时间后会尝试恢复
  - 最小请求数：熔断触发的最小请求数，请求数小于该值时即使异常比率超出阈值也不会熔断

- 异常比例：当单位统计时长内请求数目大于设置的最小请求数目，并且异常的比例大于阈值，则接下来的熔断时长内请求会自动被熔断
  - 比例阈值
  - 熔断时长：超过时间后会尝试恢复
  - 最小请求数：熔断触发的最小请求数，请求数小于该值时，即使异常比率超出阈值也不会熔断

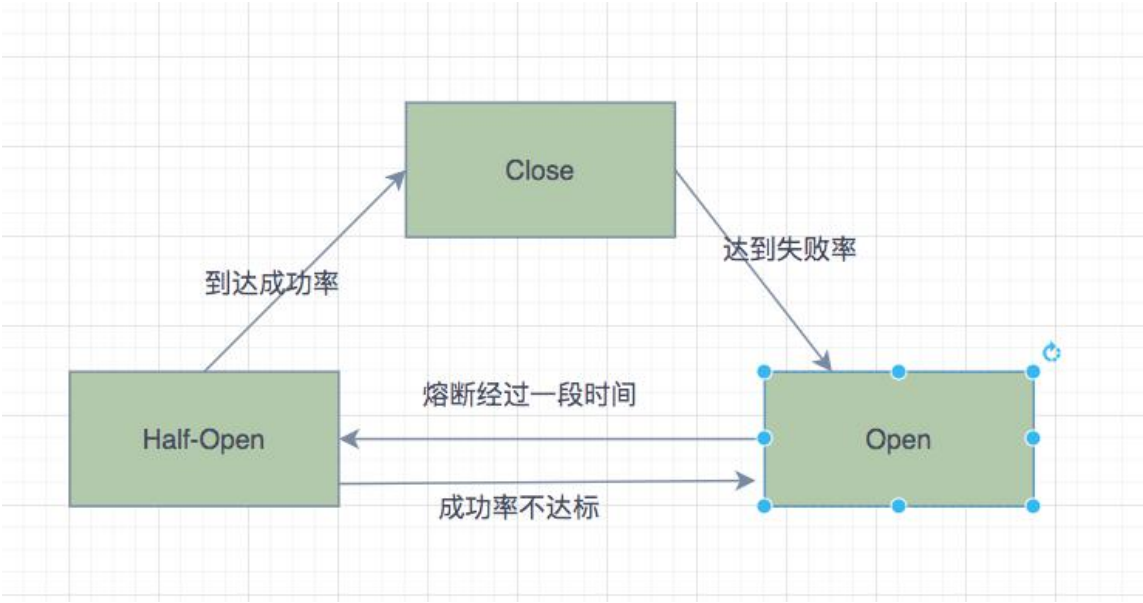
- 异常数：当单位统计时长内的异常数目超过阈值之后会自动进行熔断
  - 异常数:
  - 熔断时长：超过时间后会尝试恢复
  - 最小请求数：熔断触发的最小请求数，请求数小于该值时即使异常比率超出阈值也不会熔断

第5集 新版Sentinel的熔断状态和恢复讲解

简介：讲解服务调用常见的熔断状态和恢复

- 服务熔断一般有三种状态（画图）
  - 熔断关闭（Closed）
    - 服务没有故障时，熔断器所处的状态，对调用方的调用不做任何限制
  - 熔断开启（Open）
    - 后续对该服务接口的调用不再经过网络，直接执行本地的fallback方法
  - 半熔断（Half-Open）

    - 所谓半熔断就是尝试恢复服务调用，允许有限的流量调用该服务，并监控调用成功率



- 熔断恢复：
    - 经过熔断时长后熔断器会进入探测恢复状态（HALF-OPEN 状态）尝试恢复服务调用，允许有限的流量调用该服务，并监控调用成功率。
    - 如果成功率达到预期，则说明服务已恢复，进入熔断关闭状态；如果成功率仍旧很低，则重新进入熔断状态

第6集 新版Sentinel整合AlibabaCloud微服务熔断实操

简介：讲解服务调用熔断例子

- 修改代码
- 熔断测试

```
 int temp = 0;
    @RequestMapping("list")
    public Object list(){

//         try {
//             TimeUnit.SECONDS.sleep(3);
//         } catch (InterruptedException e) {
//             e.printStackTrace();
//         }

        temp++;
        if(temp%3 == 0){
            throw  new RuntimeException();
        }
        Map<String,String> map  = new
HashMap<>();
        map.put("title1","ALibabaCloud微服务专
题");
        map.put("title2","小滴课堂面试专题第一
季");
```

```
        return map;
    }
```

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第九章 【高级篇幅】玩转Sentinel自定义异常-整合Open-Feign

第1集 AlibabaCloud版本升级-自定义降级异常不向下兼容的坑

简介：讲解Sentinel自定义异常降级-新旧版本差异

- 默认降级返回数据问题
  - 限流和熔断返回的数据有问题-
  - 微服务交互基本都是json格式，如果让自定义异常信息
- AlibabCloud版本升级，不兼容问题

    - v2.1.0到v2.2.0后，Sentinel里面依赖进行了改动，且不向下兼容
- 自定义降级返回数据
    - 【旧版】实现UrlBlockHandler并且重写blocked方法

```
@Component
public class XdclassUrlBlockHandler
implements UrlBlockHandler {
    @Override
    public void blocked(HttpServletRequest
httpServletRequest, HttpServletResponse
httpServletResponse, BlockException e) throws
IOException {
        //降级业务处理
    }
}
```

    - 【新版】实现BlockExceptionHandler并且重写handle方法

```
public class XdclassUrlBlockHandler
implements BlockExceptionHandler {

    @Override
    public void handle(HttpServletRequest
httpServletRequest, HttpServletResponse
httpServletResponse, BlockException e) throws
Exception {
    //降级业务处理
    }
}
```

第2集 新版Sentinel自定义降级异常数据开发实战

简介：讲解新版Sentinel自定义异常数据开发实战

- 异常种类

```
FlowException   //限流异常
DegradeException   //降级异常
ParamFlowException //参数限流异常
SystemBlockException //系统负载异常
AuthorityException //授权异常
```

- 【新版】实现BlockExceptionHandler并且重写handle方法

```
@Component
public class XdclassUrlBlockHandler implements BlockExceptionHandler {
    @Override
    public void handle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, BlockException e) throws IOException {
        Map<String,Object> backMap=new HashMap<>();
        if (e instanceof FlowException){
            backMap.put("code",-1);
            backMap.put("msg","限流-异常啦");
```

```
        }else if (e instanceof
DegradeException){
            backMap.put("code",-2);
            backMap.put("msg","降级-异常啦");
        }else if (e instanceof
ParamFlowException){
            backMap.put("code",-3);
            backMap.put("msg","热点-异常啦");
        }else if (e instanceof
SystemBlockException){
            backMap.put("code",-4);
            backMap.put("msg","系统规则-异常啦");
        }else if (e instanceof
AuthorityException){
            backMap.put("code",-5);
            backMap.put("msg","认证-异常啦");
        }

        // 设置返回json数据
        httpServletResponse.setStatus(200);
        httpServletResponse.setHeader("content-
Type","application/json;charset=UTF-8");

httpServletResponse.getWriter().write(JSON.toJS
ONString(backMap));
    }
}
```

第3集 新版Sentinel整合OpenFeign配置实战

简介：使用Feign整合Sentinel配置实战

- 整合步骤
  - 加入依赖

```
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-
sentinel</artifactId>
</dependency>
```

  - 开启Feign对Sentinel的支持

```
feign:
  sentinel:
    enabled: true
```

  - 创建容错类, 实现对应的服务接口, 记得加注解

@Service

```
@Service
public class VideoServiceFallback implements
VideoService {
    @Override
    public Video findById(int videoId) {
        Video video = new Video();
        video.setTitle("熔断降级数据");
        return video;
    }

    @Override
    public Video saveVideo(Video video) {
        return null;
    }
}
```

- 配置feign容错类

```
@FeignClient(value = "xdclass-video-service",
fallback = VideoServiceFallback.class)
```

- 准备 **兜底数据**

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第十章 AlibabaCloud微服务升级下一代JDK11 LTS长期支持版本

第1集 大话JDK各个版本常见问题讲解

简介：讲解JDK一些基础知识科普

- OpenJDK和OracleJDK版本区别
  - OpenJDK是JDK的开放源码版本，以GPL协议的形式发布（General Public License）
  - Oracle JDK采用了商业实现
- LTS 是啥意思?
  - Long Term Support 长期支持的版本，如JDK8、JDK11都是属于LTS
  - JDK9 和 JDK10 这两个被称为“功能性的版本”不同, 两者

均只提供半年的技术支持
    - 甲骨文释出Java的政策，每6个月会有一个版本的释出，长期支持版本每三年发布一次，根据 后续的发布计划，下一个长期支持版 Java 17 将于2021年发布
- 8u20、11u20是啥意思?
    - 就是Java的补丁，比如JDK8的 8u20版本、8u60版本; java11的 11u20、11u40版本

第2集 AlibabaCloud微服务升级JDK11和配置

简介：讲解AlibabaCloud微服务升级JDK11

- 安装包（课程资料里面，和正常jdk安装没区别）
- IDEA配置
    - procject structure
    - 偏好设置-编译版本
- maven项目配置

```
    <properties>
        <java.version>11</java.version>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>
    </properties>
```

- 版本选择：
  - 只要不是JDK8以下就行，
  - 也不建议用最新的，比如现在JDK14或者JDK17是吧
  - 涉及到中间件的升级，不排除有些版本是把老版本的代码删除了，就GG了

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第十一章 微服务核心组件之网关讲解

第1集 什么是微服务的网关和应用场景

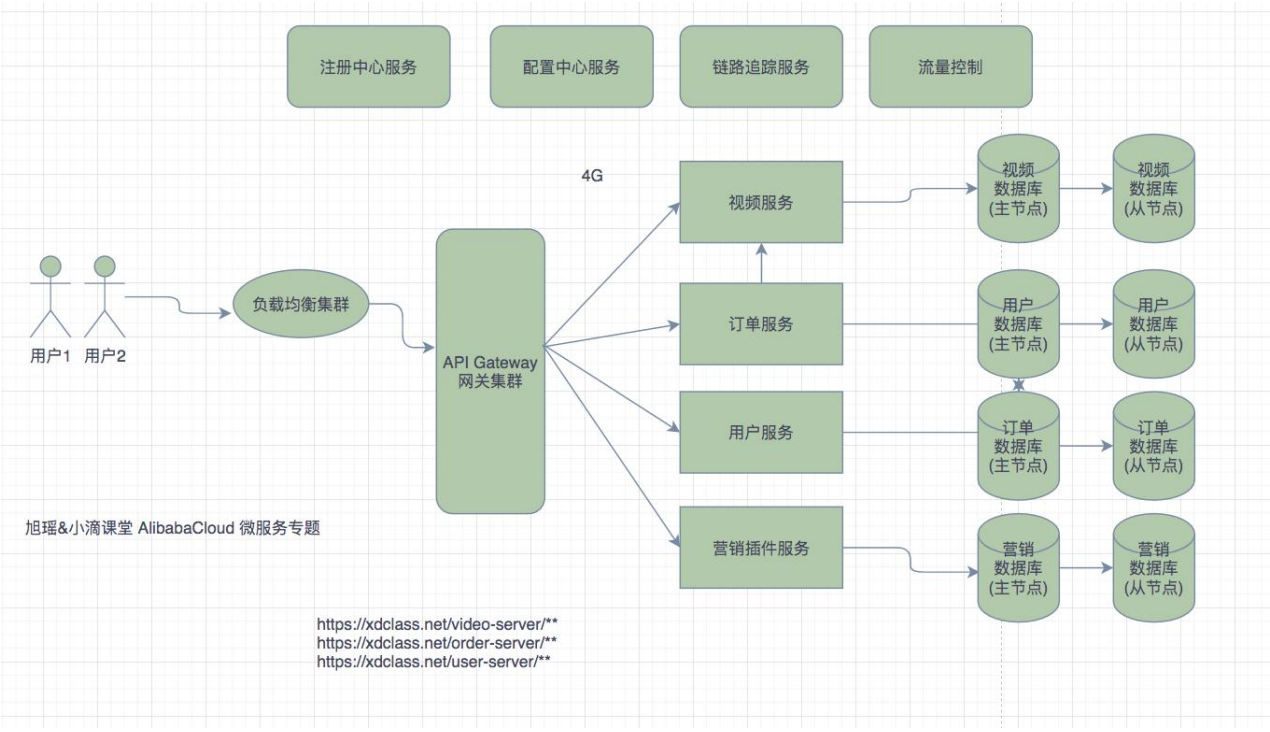
简介：介绍什么是微服务的网关和应用场景

- 什么是网关
  - API Gateway，是系统的唯一对外的入口，介于客户端和服务器端之间的中间层，处理非业务功能 提供路由请求、鉴权、监控、缓存、限流等功能
  - 统一接入
    - 智能路由
    - AB测试、灰度测试
    - 负载均衡、容灾处理
    - 日志埋点（类似Nignx日志）
  - 流量监控
    - 限流处理

        - 服务降级
    - 安全防护
        - 鉴权处理
        - 监控
        - 机器网络隔离

- 主流的网关
    - zuul：是Netflix开源的微服务网关，和Eureka,Ribbon,Hystrix等组件配合使用，依赖组件比较多，性能教差
    - kong: 由Mashape公司开源的，基于Nginx的API gateway
- nginx+lua：是一个高性能的HTTP和反向代理服务器,lua是脚本语言，让Nginx执行Lua脚本，并且高并发、非阻塞的处理各种请求
- springcloud gateway: Spring公司专门开发的网关，替代zuul

- 注意：AlibabaCloud全家桶还没对应的网关，我们就用SpringCloud官方推荐的Gateway


注册中心服务
配置中心服务
链路追踪服务
流量控制
4G
用户1
用户2
负载均衡集群
API Gateway
网关集群
视频服务
订单服务
用户服务
营销插件服务
视频
数据库
(主节点)
视频
数据库
(从节点)
用户
数据库
(主节点)
用户
数据库
(从节点)
订单
数据库
(主节点)
订单
数据库
(从节点)
营销
数据库
(主节点)
营销
数据库
(从节点)
旭瑶&小滴课堂 AlibabaCloud 微服务专题
https://xdclass.net/video-server/**
https://xdclass.net/order-server/**
https://xdclass.net/user-server/**

第2集 微服务的网关SpringCloud Gateway介绍

简介：介绍网关SpringCloud Gateway

- 什么是 SpringCloud Gateway
  - Spring官方出品，基于Spring5+Reactor技术开发的网关
  - 性能强劲基于Reactor+WebFlux、功能多样
  - 基于springboot2.x, 直接可以jar包方式运行
- 官方文档
  - https://spring.io/projects/spring-cloud-gateway

第3集 SpringCloud Gateway项目创建和依赖添加

简介：创建SpringCloud网关项目和依赖添加

- 创建Gateway项目
  - 添加依赖

```
 <dependency>

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

  - 配置实战

```
server:
  port: 8888
spring:
  application:
    name: api-gateway
  cloud:
    gateway:
      routes: #数组形式
        - id: order-service  #路由唯一标识
          uri: http://127.0.0.1:8000  #想要转
发到的地址
```

```
            order: 1 #优先级，数字越小优先级越高
            predicates: #断言 配置哪个路径才转发
              - Path=/order-server/**
            filters: #过滤器，请求在传递过程中通过过滤器修改
              - StripPrefix=1  #去掉第一层前缀

#访问路径 http://localhost:8888/order-server/api/v1/video_order/list
#转发路径 http://localhost:8000/order-server/api/v1/video_order/list
#需要过滤器去掉前面第一层
```

- 配置项怎么看?
  - 点击routes进去

第4集 SpringCloud Gateway网关整合Nacos开发实战

简介：讲解Gateway配置Nocas和实战

- 原先存在的问题

    - 微服务地址写死
    - 负载均衡没做到
- 添加Nacos服务治理配置
    - 网关添加naocs依赖

```
        <!--添加nacos客户端-->
        <dependency>

<groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-
alibaba-nacos-discovery</artifactId>
        </dependency>
```

    - 启动类开启支持

```
@EnableDiscoveryClient
```

    - 修改配置文件

```
server:
  port: 8888
spring:
  application:
    name: api-gateway
  cloud:
    nacos:
      discovery:
        server-addr: 127.0.0.1:8848
```

```
    gateway:
      routes: #数组形式
        - id: order-service  #路由唯一标识
          #uri: http://127.0.0.1:8000  #想要转发到的地址
          uri: lb://xdclass-order-service  #从nacos获取名称转发,lb是负载均衡轮训策略

          predicates: #断言 配置哪个路径才转发
            - Path=/order-server/**
          filters: #过滤器，请求在传递过程中通过过滤器修改
            - StripPrefix=1 #去掉第一层前缀
      discovery:
        locator:
          enabled: true  #开启网关拉取nacos的服务


## 访问路径 http://localhost:8888/order-server/api/v1/video_order/list
```

第十二章 【进阶篇】网关Gateway架构+断言+过滤器进阶实战

第1集 进阶掌握 SpringCloud Gateway配置和交互流程

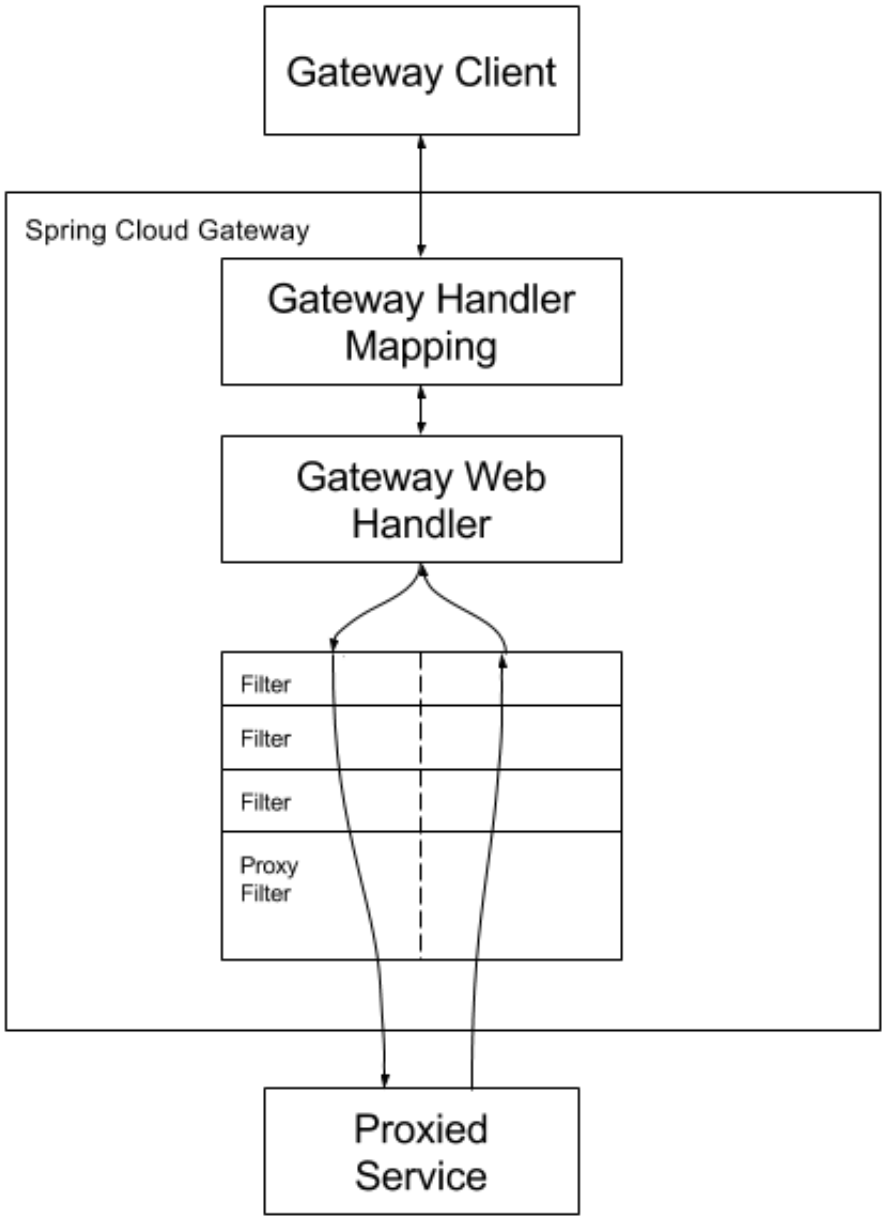
简介：讲解SpringCloud Gateway架构流程

- 网关的配置项回顾
  - 路由：是网关的基本单元，由ID、URI、一组Predicate、一组Filter组成，根据Predicate进行匹配转发

```
route组成部分
id：路由的ID
uri：匹配路由的转发地址
predicates：配置该路由的断言，通过PredicateDefinition类进行接收配置。
order：路由的优先级，数字越小，优先级越高。
```

- 交互流程

- 客户端向Spring Cloud Gateway发出请求
- 如果网关处理程序映射确定请求与路由匹配
- 则将其发送到网关Web处理程序
- 通过特定过滤器链运行，前置处理-后置处理

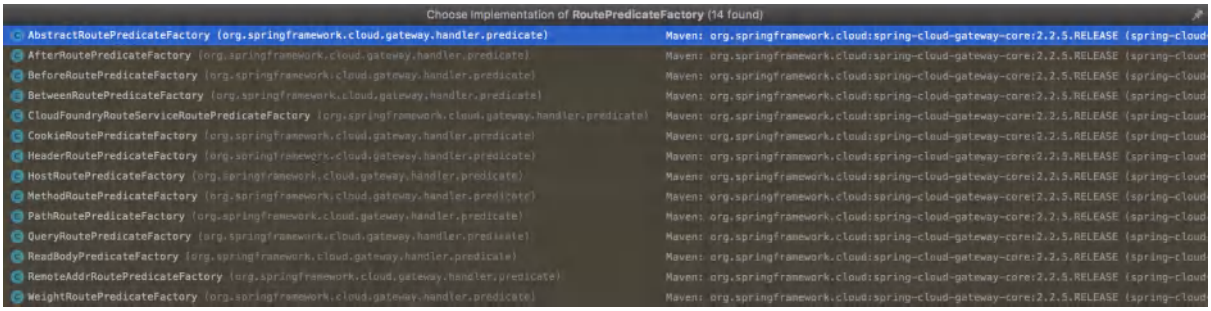


第2集 微服务SpringCloud Gateway内置路由断言讲解

简介：讲解Gateway内置的路由断言

- 什么是Gateway路由断言
  - Predicate 来源于Java8，接受输入参数，返回一个布尔值结果
  - Spring Cloud Gateway 中 Spring 利用 Predicate 的特性实现了各种路由匹配规则
  - 转发的判断条件，SpringCloud Gateway支持多种方式，常见如：Path、Query、Method、Header等
  - 支持多个`Predicate`请求的转发是必须满足所有的`Predicate`后才可以进行路由转发

- 内置路由断言介绍 RoutePredicateFactory 接口实现类



- 参数编写规则 XXXRoutePredicateFactory，使用XXX作为参数配置, 例如下面

```
predicates:
  - Host=
  - Path=
  - Method=
  - Header=
  - Query=
  - Cookie=
```

第3集 Gateway内置断言实现接口定时下线实战

简介：Gateway内置的路由接口定时下线实战

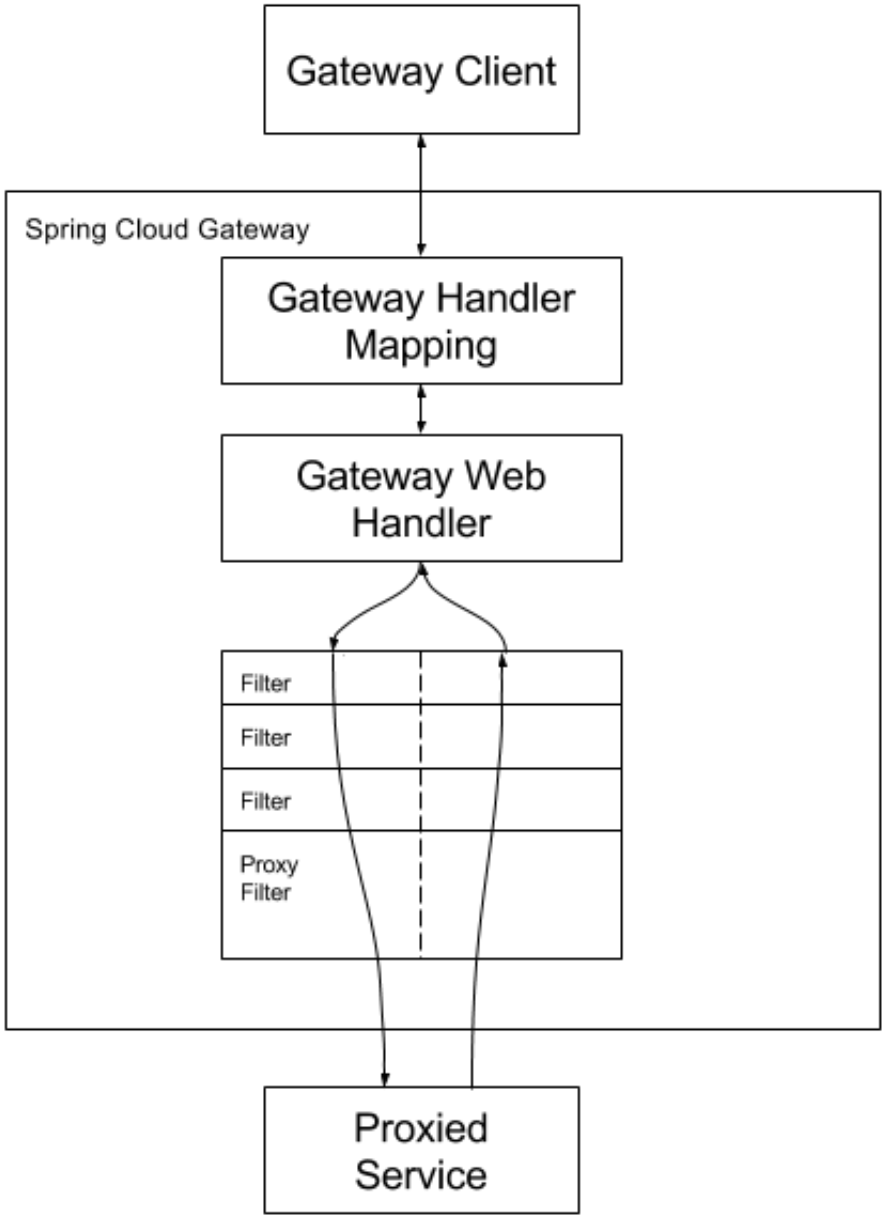
- 需求：接口需要在指定时间进行下线，过后不可以在被访问
  - 使用Before ,只要当前时间小于设定时间，路由才会匹配请求
  - 东8区的2020-09-11T01:01:01.000+08:00后，请求不可访问
  - 为了方便测试，修改时间即可

```
predicates:
  - Before=2020-09-09T01:01:01.000+08:00
```

第4集 玩转SpringCloud Gateway过滤器

简介：讲解Gateway过滤器

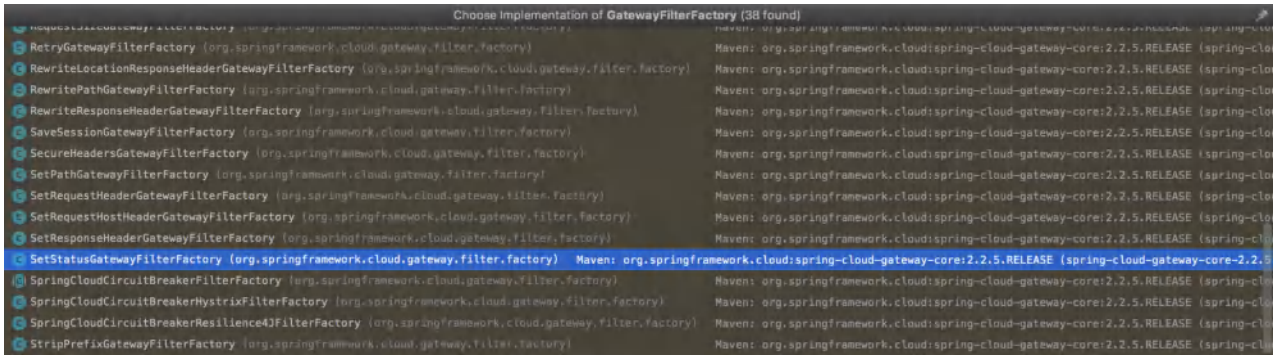
- 什么是网关的过滤器



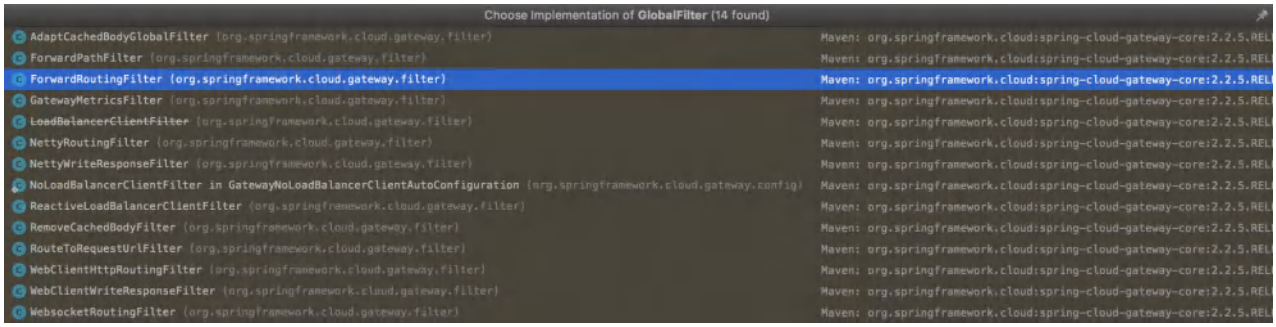
- 过滤器生命周期
  - PRE： 这种过滤器在请求被路由之前调用,一般用于鉴权、限流等
  - POST：这种过滤器在路由到微服务以后执行，一般用于修改响应结果，比如增加header信息、打点结果日志

- 网关过滤器分类

    - 局部过滤器GatewayFilter：应用在某个路由上,每个过滤器工厂都对应一个实现类，并且这些类的名称必须以 GatewayFilterFactory 结尾
    - 全局过滤器：作用全部路由上,
- 内置很多局部过滤器，顶级接口 GatewayFilterFactory,



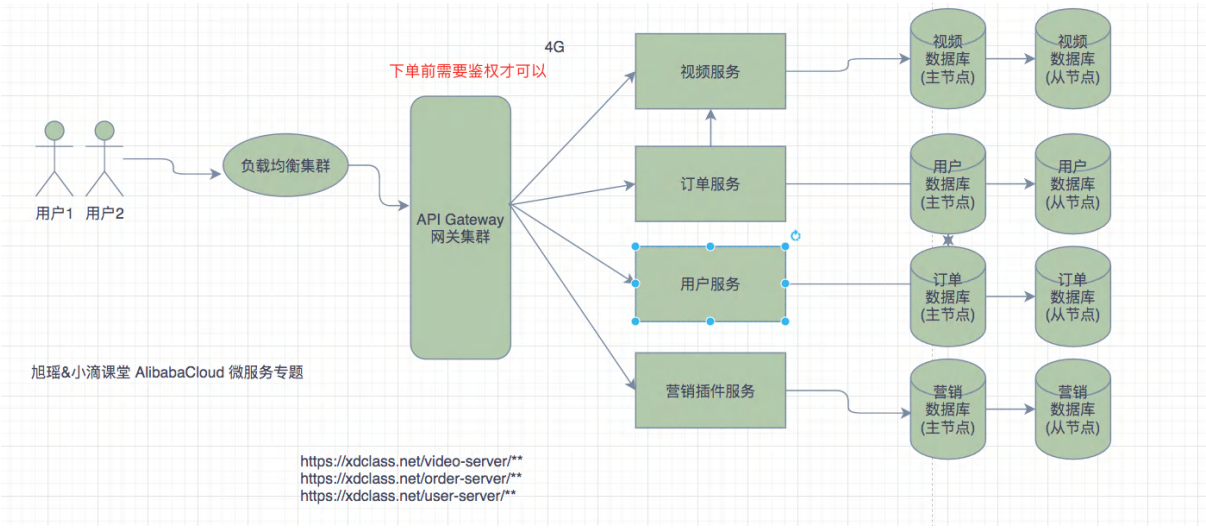
- 内置很多全局过滤器，顶级接口 GlobalFilter



第5集 【高级篇】案例实战之网关Gateway全局过滤器实现用户鉴权

简介：讲解Gateway全局过滤器实现用户鉴权

- 业务流程



- 自定义全局过滤器实现鉴权

```
@Component
public class UserGlobalFilter implements
GlobalFilter,Ordered {
```

```
    @Override
    public Mono<Void> filter(ServerWebExchange
exchange, GatewayFilterChain chain) {

        String token =
exchange.getRequest().getHeaders().getFirst("to
ken");

        System.out.println(token);
        if(StringUtils.isBlank(token)){

exchange.getResponse().setStatusCode(HttpStatus
.UNAUTHORIZED);
            return
exchange.getResponse().setComplete();
        }

        //继续往下执行
        return chain.filter(exchange);

    }

    //数字越小，优先级越高
    @Override
    public int getOrder() {
        return 0;
    }
}
```

- 路径： http://localhost:8888/order-server/api/v1/video_order/list?source=wechat
- 注意：网关不要加太多业务逻辑，否则会影响性能，务必记住

第十三章 AlibabaCloud微服务下的链路追踪系统实战

第1集 微服务架构下的排查问题复杂性概述

简介：讲解微服务链路追踪系统的作用

- 抛两个常见的问题
  - 微服务调用链路出现了问题怎么快速排查?
- 微服务调用链路耗时长怎么定位是哪个服务?
- 链路追踪系统
  - 分布式应用架构虽然满足了应用横向扩展的需求，但是运维和诊断的过程变得越来越复杂，例如会遇到接口诊断困难、应用性能诊断复杂、架构分析复杂等难题，传统的监控工具并无法满足，分布式链路系统由此诞生
- 核心：将一次请求分布式调用，使用GPS定位串起来，记录每个调用的耗时、性能等日志，并通过可视化工具展示出来
- 注意：AlibabaCloud全家桶还没对应的链路追踪系统，我们使用Sleuth和zipking（内部使用的鹰眼）

第2集 SpringCloud的链路追踪组件Sleuth实战

简介：讲解什么Sleuth链路追踪系统

- 什么是Sleuth
  - 一个组件，专门用于记录链路数据的开源组件
  - 文档：https://spring.io/projects/spring-cloud-sleuth
  - 案例

```
    [order-
service,96f95a0dd81fe3ab,852ef4cfcdecabf3,false
]

    第一个值，spring.application.name的值

    第二个值，96f95a0dd81fe3ab ，sleuth生成的一个ID，叫Trace ID，用来标识一条请求链路，一条请求链路中包含一个Trace ID，多个Span ID

    第三个值，852ef4cfcdecabf3、spanid 基本的工作单元，获取元数据，如发送一个http

    第四个值：false，是否要将该信息输出到zipkin服务中来收集和展示。
```

- 各个微服务添加依赖

```
<dependency>

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

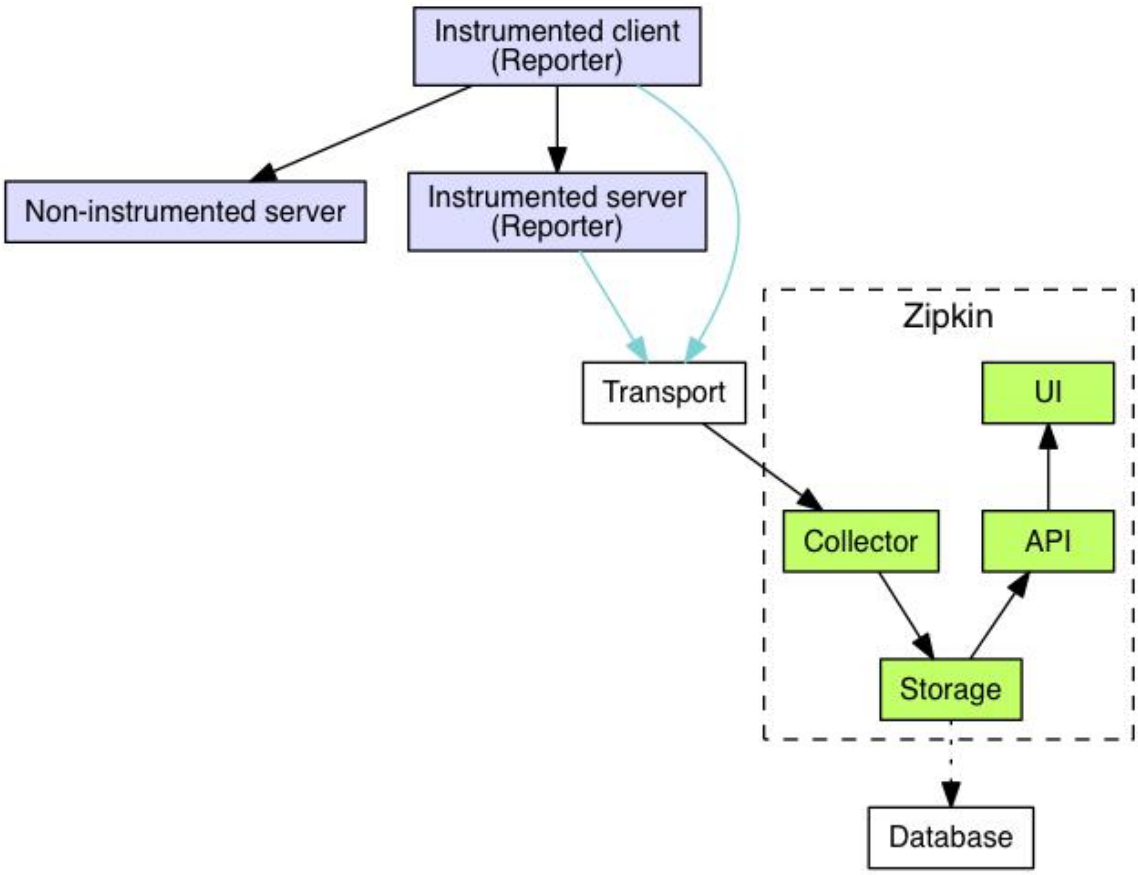
第3集 微服务下的可视化链路追踪系统Zipkin实战

简介：讲解zipkin介绍和部署实战

- 什么是zipkin
  - 官网
    - https://zipkin.io/
    - https://zipkin.io/pages/quickstart.html
  - 大规模分布式系统的APM工具（Application Performance Management）,基于Google Dapper的基础实现，和sleuth结合可以提供可视化web界面分析调用链路耗时情况
- 同类产品
  - 鹰眼（EagleEye）
  - CAT
  - twitter开源zipkin，结合sleuth
  - Pinpoint，运用JavaAgent字节码增强技术
- StackDriver Trace (Google)
- 开始使用
  - 安装包在资料里面，启动服务

```
java -jar zipkin-server-2.12.9-exec.jar
```

- 访问入口：http://127.0.0.1:9411/zipkin/
- zipkin组成：Collector、Storage、Restful API、Web UI组成



第4集 【高级篇幅】链路追踪组件 Zipkin+Sleuth整合实战

简介：使用Zipkin+Sleuth业务分析调用链路分析实战

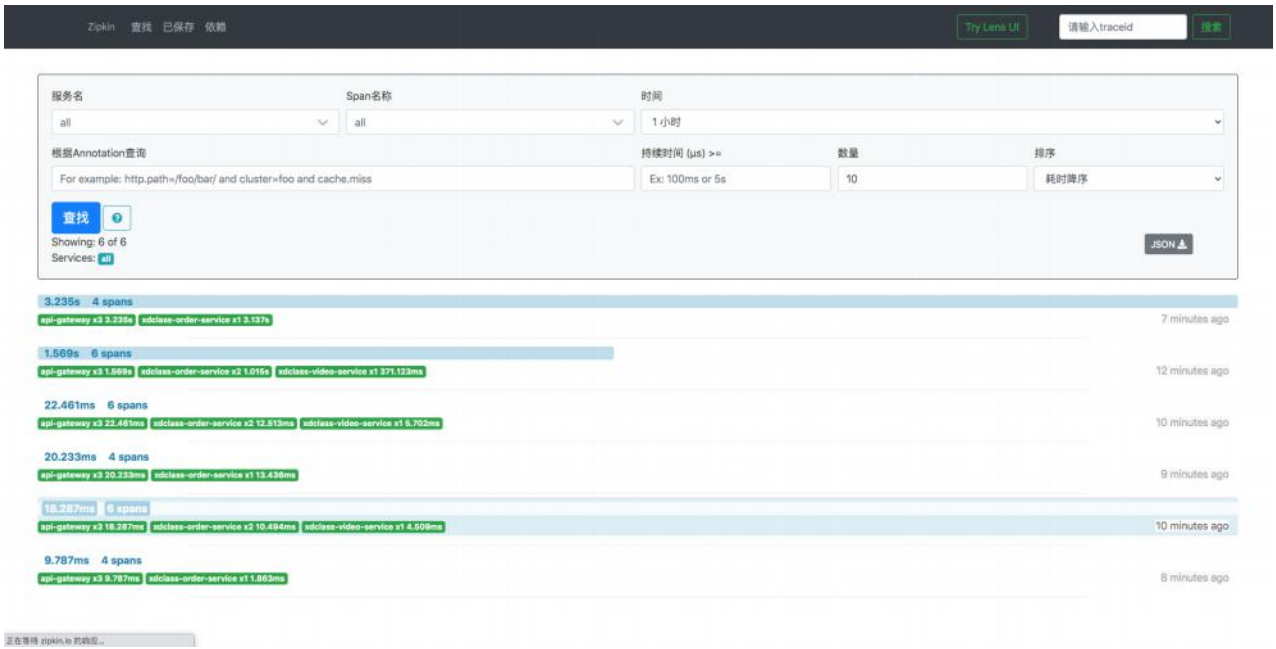
- sleuth收集跟踪信息通过http请求发送给zipkin server
- zipkin server进行跟踪信息的存储以及提供Rest API即可
- Zipkin UI调用其API接口进行数据展示默认存储是内存，可也用mysql 或者elasticsearch等存储
- 微服务加入依赖

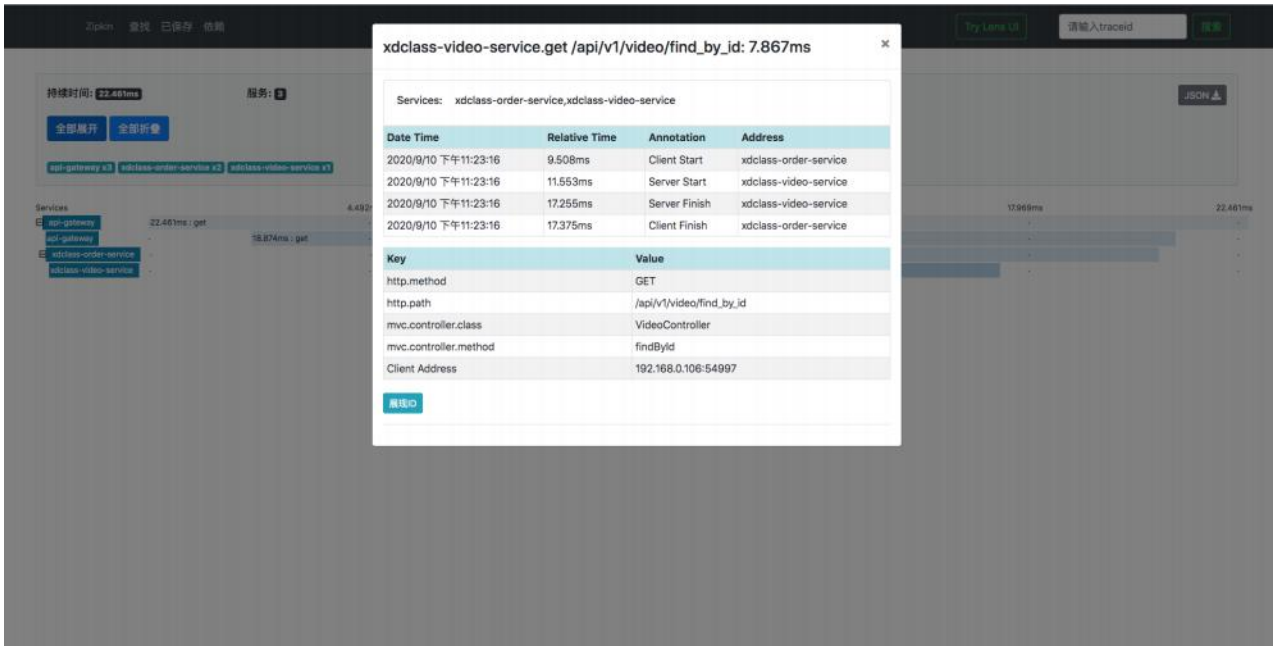
```
<dependency>

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

- 配置地址和采样百分比配置

```
spring:
  application:
    name: api-gateway
  zipkin:
    base-url: http://127.0.0.1:9411/ #zipkin地址
    discovery-client-enabled: false  #不用开启服务发现

  sleuth:
    sampler:
      probability: 1.0 #采样百分比
```

默认为0.1，即10%，这里配置1，是记录全部的sleuth信息，是为了收集到更多的数据（仅供测试用）。
在分布式系统中，过于频繁的采样会影响系统性能，所以这里配置需要采用一个合适的值。




xdclass-video-service.get /api/v1/video/find_by_id: 7.867ms
Services: xdclass-order-service,xdclass-video-service
Date Time | Relative Time | Annotation | Address
2020/9/10 下午11:23:16 | 9.508ms | Client Start | xdclass-order-service
2020/9/10 下午11:23:16 | 11.553ms | Server Start | xdclass-video-service
2020/9/10 下午11:23:16 | 17.255ms | Server Finish | xdclass-video-service
2020/9/10 下午11:23:16 | 17.375ms | Client Finish | xdclass-order-service
Key | Value
http.method | GET
http.path | /api/v1/video/find_by_id
mvc.controller.class | VideoController
mvc.controller.method | findById
Client Address | 192.168.0.106:54997

第5集 微服务链路追踪系统Zipkin持久化配置

简介：实战zipkin+sleuth链路追踪日志持久化

- 现存在的问题
  - 服务重启会导致链路追踪系统数据丢失
- 持久化配置：mysql或者elasticsearch
  - 创建数据库表SQL脚本 (本章本集的资料里面, 不要说找不到)
  - 启动命令

```
java -jar zipkin-server-2.12.9-exec.jar --
STORAGE_TYPE=mysql --MYSQL_HOST=127.0.0.1 --
MYSQL_TCP_PORT=3306 --MYSQL_DB=zipkin_log --
MYSQL_USER=root --MYSQL_PASS=xdclass.net
```

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第十四章 AlibabaCloud微服务下的分布式配置中心实战

第1集 微服务下的分布式配置中心

简介：讲解什么是配置中心及使用前后的好处

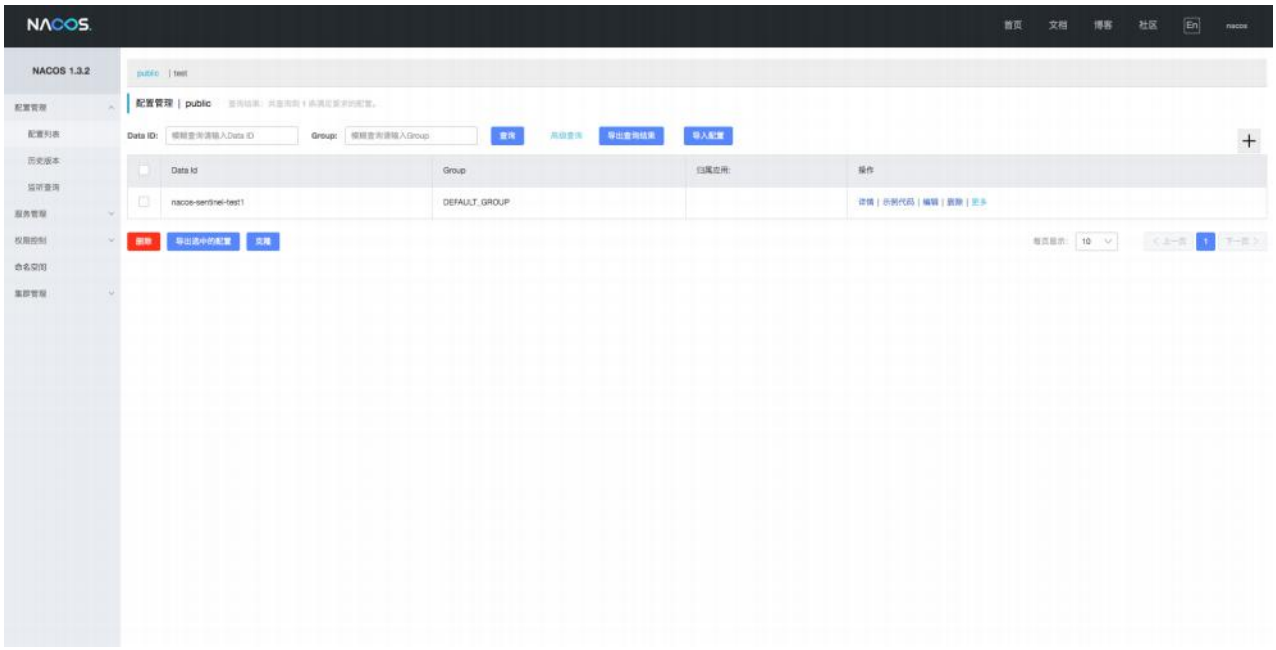
- 现在微服务存在的问题
  - 配置文件增多，不好维护
  - 修改配置文件需要重新发布
- 什么是配置中心：
  - 一句话：统一管理配置, 快速切换各个环境的配置
  - 相关产品：

- 百度的disconf 地址:https://github.com/knightliao/disconf
- 阿里的diamand 地址：https://github.com/takeseem/diamond
- springcloud的configs-server: 地址：http://cloud.spring.io/spring-cloud-config/
- 阿里的Nacos:既可以当服务治理，又可以当配置中心，Nacos = Eureka + Config

第2集 AlibabaCloud配置中心Nacos面板介绍

简介：讲解Nacos作为配置中心面板介绍

- Nacos配置中心面板介绍



- 项目添加依赖

```
 <dependency>

<groupId>com.alibaba.cloud</groupId>
            <artifactId>spring-cloud-starter-
alibaba-nacos-config</artifactId>
</dependency>
```

- 官方文档
  - https://github.com/alibaba/spring-cloud-alibaba/wiki/Nacos-config

第3集 新版AlibabaCloud配置中心Nacos实战和注意的坑

简介：讲解Nacos作为配置中心实战

- 配置文件优先级讲解
    - 不能使用原先的application.yml, 需要使用bootstrap.yml作为配置文件

    - 配置读取优先级 bootstrap.yml > application.yml
- 配置实操
    - 订单服务迁移配置
    - 增加bootstrap.yml

```
spring:
  application:
    name: xdclass-order-service
  cloud:
    nacos:
      config:
        server-addr: 127.0.0.1:8848 #Nacos配
置中心地址
        file-extension: yaml #文件拓展格式

  profiles:
    active: dev
```

- 启动微服务服务验证
    - 测试是否可以获取配置

浏览器访问
http://127.0.0.1:8848/nacos/v1/cs/configs?dataId=xdclass-order-service-dev.yaml&group=DEFAULT_GROUP

部分同学如果出现 config dta not exist 建议重启nacos

- 除开上述问题，如果还是拉取不到配置(保持和课程版本，文件名一样先)

重新构建下项目

```
mvn clean package -U
```

然后重启IDEA

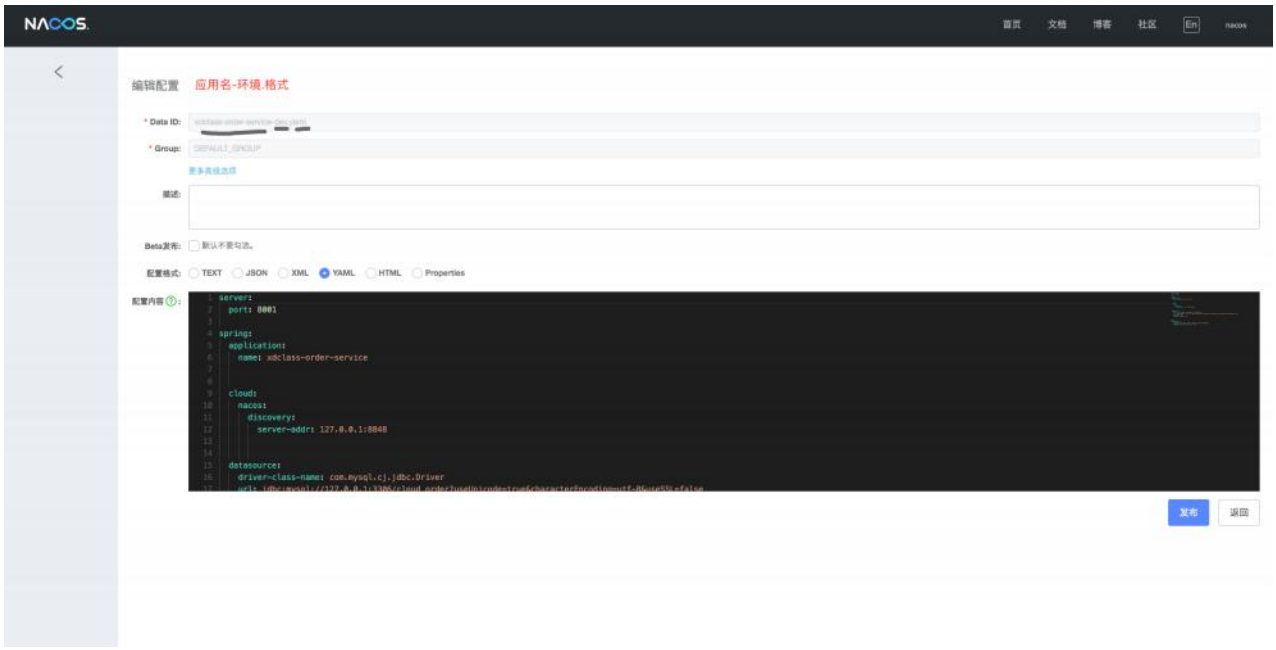
- dataId组成，在 Nacos Spring Cloud 中，dataId 的完整格式如下

${prefix}-${spring.profiles.active}.${file-extension}

prefix 默认为 spring.application.name 的值

spring.profiles.active 即为当前环境对应的 profile 当 spring.profiles.active 为空时，对应的连接符 - 也将不存在，dataId 的拼接格式变成 ${prefix}.${file-extension}

file-exetension 为配置内容的数据格式，可以通过配置项 spring.cloud.nacos.config.file-extension 来配置。目前只支持 properties 和 yaml 类型。



第4集 【高级篇】AlibabaCloud配置中心Nacos动态配置下发实战

简介：讲解Nacos动态刷新配置

- 什么是动态刷新配置
  - 我们修改了配置，程序不能自动更新
  - 动态刷新就可以解决这个问题
- 配置实战
  - 增加Nacos增加测试配置
  - 编写代码

```
@RefreshScope
public class OrderController {
    @Value("${video.title}")
    private String videoTitle;
}
```

愿景："让编程不再难学，让技术与生活更加有趣" 更多架构课程请访问 xdclass.net

第十五章 阿里云ECS服务器介绍和网络知识讲解

第1集 云服务器介绍和阿里云服务器ECS服务器选购

简介：什么是云服务器及目前主要的几个厂商介绍

- 云厂商
  - 阿里云：https://www.aliyun.com/
  - 腾讯云：https://cloud.tencent.com/
  - 亚马逊云：https://aws.amazon.com/
  - 阿里云新用户地址（如果地址失效，联系我或者客服即可，1折）
    - https://www.aliyun.com/minisite/goods?userCode=r5saexap&share_source=copy_link
- 环境问题说明
  - Win7、Win8、Win10、Mac、虚拟机等等，可能存在兼容问题
  - 务必使用CentOS 7 以上版本，64位系统！！！！
  - 选购实操

第2集 阿里云服务器远程登录和常用工具

简介：讲解阿里云服务器登录使用和常见终端工具

- 备注：(服务器、域名等使用你们自己购买的哈，上面有提供低价购买链接，失效找我)
- 阿里云新用户地址 https://www.aliyun.com/minisite/goods?userCode=r5saexap&share_source=copy_link
- 控制台修改阿里云远程连接密码
- windows工具 putty，xshell, security CRT
    - 参考资料：
        - https://jingyan.baidu.com/article/e75057f210c6dcebc91a89dd.html

        - https://www.jb51.net/softjc/88235.html
- 苹果系统MAC： 通过终端登录
    - ssh root@ip 回车后输入密码
    - ssh root@120.24.216.117
- linux图形操作工具（用于远程连接上传文件）
    - mac: filezilla
        - sftp://120.24.216.117
    - windows: winscp
    - 参考资料：https://jingyan.baidu.com/article/ed2a5d1f346fd409f6be179a.html
- 可以尝试自己通过百度进行找文档， 安装mysql jdk nginx maven git redis等，也可以看我们的课程

第3集 域名备案和线上访问服务器应用流程解析

简介：讲解应用部署到可以公网访问需要的知识

- 一个http请求基本流程
  - 客户端通过发起域名资源请求 -> DNS解析获得IP -> 寻找服务器获得资源
- 域名和ip的关系，DNS作用
  - DNS：Domain Name Server 域名服务器 域名虽然便于人们记忆，但网络中的计算机之间只能互相认识IP地址，它们之间的转换工作称为域名解析，域名解析需要由专门的域名解析服务器来完成，DNS 就是进行域名解析的服务器
- 什么是cname和a记录
  - a记录
    - 用户可以在此设置域名并指向到自己的目标主机地址上，从而实现通过域名找到服务器（也叫ip指向域名配置）aabbcc.com -> 120.24.216.117
  - cname
    - 别名指向，可以为一个主机设置别名。比如设置open1024.com，用来指向一个主机 xdclass.net 那么以后就可以用open1024.com来代替访问xdclass.net 了
    - www.xdclass.net --> xdclass.net

- 购买服务器，阿里云，腾讯云，亚马逊云aws
- 购买域名，备案
    - 阿里云 备案地址：https://beian.aliyun.com/
- 安装项目依赖的基本环境，比如java、nginx等软件(看项目)
- 【注意】
    - 备案需要1~2个工作日，推荐使用这个方式，掌握上线流程
    - 实在不行，则去网上找虚拟机教程，本地搭建虚拟机，然后看下一章视频进行配置

第十六章 微服务容器化部署Docker专题

第1集 分布式架构-微服务下的Docker介绍和使用场景

简介：Docker介绍和使用场景

- 官网：https://www.docker.com/get-started
- 什么是Dokcer
  - 百科:一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上，也可以实现虚拟化。
  - 容器是完全使用沙箱机制，相互之间不会有任何接口，使用go语言编写，在LCX（linux容器）基础上进行的封装
  - 简单来说：
    - 就是可以快速部署启动应用
    - 实现虚拟化，完整资源隔离
    - 一次编写，四处运行
    - 但有一定的限制，比如Docker是基于Linux 64bit的，无法在32bit的linux/Windows/unix环境下使用

- 为什么要用

- 提供一次性的环境，假如需要安装Mysql，则需要安装很多依赖库、版本等，如果使用Docker则通过镜像就可以直接启动运行
- 快速动态扩容，使用docker部署了一个应用，可以制作成镜像，然后通过Dokcer快速启动
- 组建微服务架构，可以在一个机器上模拟出多个微服务，启动多个应用
- 更好的资源隔离和共享
- 一句话：开箱即用，快速部署，可移植性强，环境隔离

第2集 阿里云Linux云服务器Centos 64位安装Docker实战

简介：讲解阿里云ECS服务安装Docker实战

- 远程连接ECS实例
- 依次运行以下命令添加yum源。

```
yum update
yum install epel-release -y
yum clean all
yum list
```

- 安装并运行Docker。

```
yum install docker-io -y
systemctl start docker
```

- 检查安装结果。

```
docker info
```

- 启动使用Docker

```
systemctl start docker      #运行Docker守护进程
systemctl stop docker       #停止Docker守护进程
systemctl restart docker    #重启Docker守护进程
```

- 更多文档

```
https://help.aliyun.com/document_detail/51853.html?spm=a2c4g.11186623.6.820.RaToNY
```

第3集 面试对象的方式快速掌握 Docker仓库、镜像、容器核心概念

简介：快速掌握Dokcer基础知识

- 概念：
  - Docker 镜像 - Docker images：容器运行的只读模板，操作系统+软件运行环境+用户程序

```
class User{
 private String userName;
 private int age;
}
```

  - Docker 容器 - Docker containers：容器包含了某个应用运行所需要的全部环境

```
 User user = new User()
```

- Docker 仓库 - Docker registeries： 用来保存镜像，有公有和私有仓库，好比Maven的中央仓库和本地私服
- 总结 对比面向对象的方式

```
Dokcer 里面的镜像 ： Java里面的类 Class
Docker 里面的容器 ： Java里面的对象 Object
通过类创建对象，通过镜像创建容器
```

第4集 玩转Docker容器常见命令实战

简介：掌握Docker容器常见命令

- 常用命令（安装部署好Docker后，执行的命令是docker开头）,xxx是镜像名称
- 搜索镜像：docker search xxx
- 列出当前系统存在的镜像：docker images
- 拉取镜像：docker pull xxx

    - xxx是具体某个镜像名称(格式 REPOSITORY:TAG)
    - REPOSITORY：表示镜像的仓库源,TAG：镜像的标签
- 运行一个容器：

```
docker run --name nginx-xd -p 8080:80 -d
nginx

docker run - 运行一个容器
      -d 后台运行
      -p 端口映射
      --name "xxx"  容器名称
```

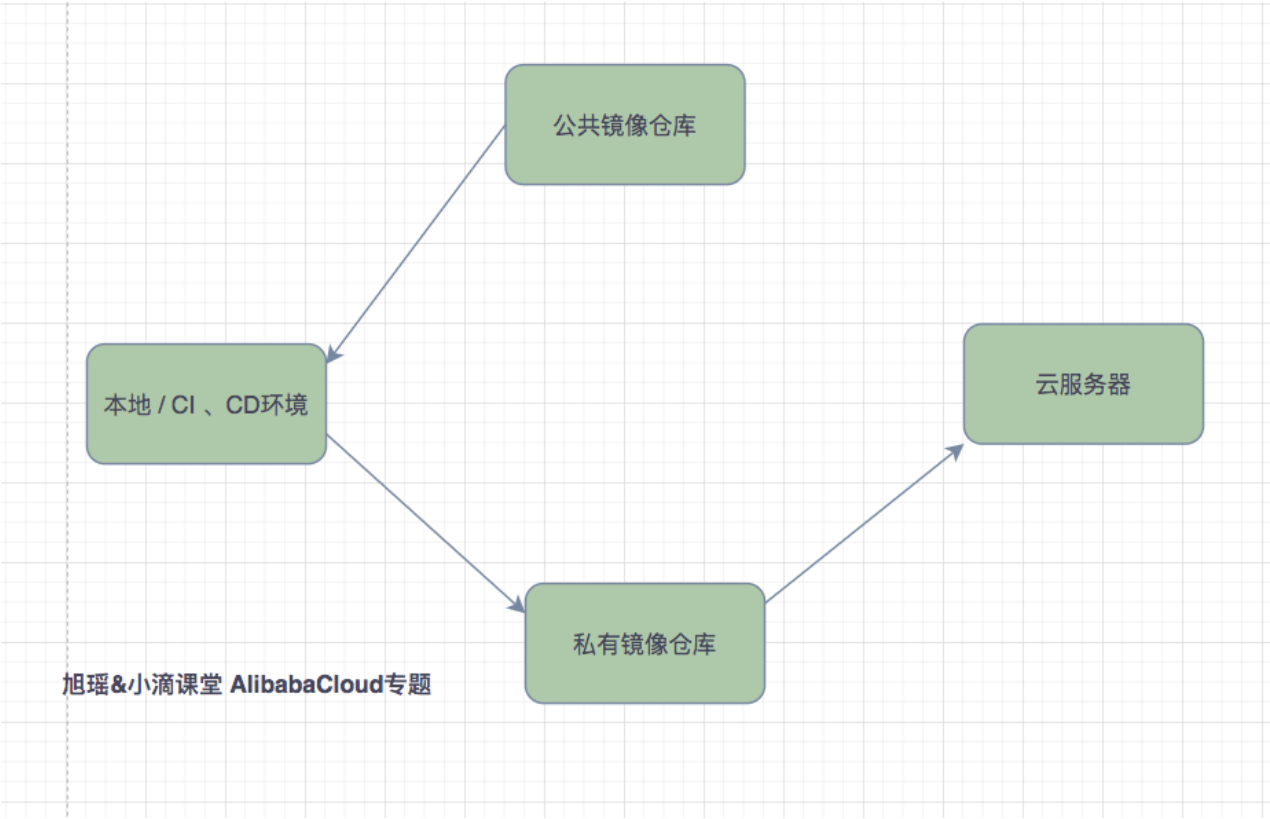
- 列举当前运行的容器：docker ps
- 检查容器内部信息：docker inspect 容器名称
- 删除镜像：docker rmi IMAGE_NAME
    - 强制移除镜像不管是否有容器使用该镜像 增加 -f 参数
- 停止某个容器：docker stop 容器名称
- 启动某个容器：docker start 容器名称
- 移除某个容器： docker rm 容器名称 （容器必须是停止状态）
- 列举全部 容器： docker ps -a

第5集 容器化部署必备Docker公有+阿里云私有镜像仓库

简介：掌握Docker仓库的知识

- 为啥要用镜像仓库
- 官方公共镜像仓库和私有镜像仓库(画图)
  - 公共镜像仓库：
    - 官方：https://hub.docker.com/，基于各个软件开发或者有软件提供商开发的
    - 非官方：其他组织或者公司开发的镜像，供大家免费试用
  - 私有镜像仓库：
    - 用于存放公司内部的镜像，不提供给外部试用；
- 开通阿里云私有镜像仓库
  - 登录阿里云账号访问地址：
    - https://cr.console.aliyun.com/
    - https://cr.console.aliyun.com/cn-shenzhen/instances/credentials

- 初次使用会提示开通



第6集 不同系统Docker安装常见问题讲解和解决思路

简介：常见系统安装Docker和一些坑

- 本地需要安装Docker,才可以进行打包，搭建
  - Win7~Win10
  - Mac
  - Linux
    - CentOS
    - ubuntu
  - 官方地址
    - https://docs.docker.com/docker-for-mac/install/
    - https://docs.docker.com/docker-for-windows/install/
- 问题
  - 镜像下载慢
    - 搜索修改镜像仓库地址：阿里云、网易云等都有镜像仓库地址（不熟悉不建议乱修改）
  - 根据打包错误日志搜索博文
  - 本地时间不同步
  - 本地网络差，下载包容易超时或者慢（只能等）

- 常规的部署只是Docker的冰山一角，课程快速入门，如果想深入可以看我们的Docker专题

- 查看容器启动日志
    - docker logs -f containerid

第十七章 【高级篇】新版JDK11+AlibabaCloud+Docker整合打包镜像推送

第1集 新版AlibabaCloud微服务本地镜像打包配置讲解

简介：微服务Docker镜像打包讲解

- 官方文档：https://spring.io/guides/gs/spring-boot-docker/
- 父项目怎么springboot版本依赖

```
 <properties>
        <java.version>11</java.version>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<spring.boot.version>2.3.3.RELEASE</spring.boot.version>
  </properties>
```

- 每个子模块项目添加依赖

```
//配置文件增加
<docker.image.prefix>xdclass</docker.image.pref
ix>


<build>
        <finalName>alibaba-cloud-
gateway</finalName>
        <plugins>
            <plugin>

<groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-
plugin</artifactId>

                <configuration>
                    <fork>true</fork>

<addResources>true</addResources>
                </configuration>
            </plugin>

            <plugin>
                <groupId>com.spotify</groupId>
                <artifactId>dockerfile-maven-
plugin</artifactId>
                <version>1.4.10</version>
```

```
                <configuration>

<repository>${docker.image.prefix}/${project.artifactId}</repository>
                    <buildArgs>

<JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
                    </buildArgs>
                </configuration>
            </plugin>
        </plugins>
    </build>
```

- Spotify 的 docker-maven-plugin 插件是用maven插件方式构建docker镜像的。

```
${project.build.finalName} 产出物名称，缺省为${project.artifactId}-${project.version}
```

第2集 镜像打包脚本Dockerfile介绍和【新版JDK11】整合

简介：微服务Docker镜像打包整合JDK11

- 创建Dockerfile,默认是根目录，（可以修改为src/main/docker/Dockerfile,如果修则需要制定路径）
- 什么是Dockerfile
  - 由一系列命令和参数构成的脚本，这些命令应用于基础镜像, 最终创建一个新的镜像

```
FROM  adoptopenjdk/openjdk11:ubi
VOLUME /tmp
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

FROM <image>:<tag> 需要一个基础镜像，可以是公共的或者是私有的，
后续构建会基于此镜像，如果同一个Dockerfile中建立多个镜像时，可以使用多个FROM指令

VOLUME 配置一个具有持久化功能的目录，主机/var/lib/docker 目录下创建了一个临时文件，并链接到容器的/tmp。改步骤是可选的，如果涉及到文件系统的应用就很有必要了。

```
/tmp目录用来持久化到 Docker 数据文件夹，因为
Spring Boot 使用的内嵌 Tomcat 容器默认使用/tmp作
为工作目录

ARG  设置编译镜像时加入的参数， JAR_FILE 是设置容器
的环境变量(maven里面配置的)
COPY ： 只支持将本地文件复制到容器 ,还有个ADD更强大
但复杂点
ENTRYPOINT 容器启动时执行的命令

EXPOSE 8080 暴露镜像端口
```

- 构建镜像(去到子模块pom文件下)

```
mvn install -Dmaven.test.skip=true
dockerfile:build
```

- 打包网关镜像

第3集 AlibabaCloud-视频服务-订单服务Docker镜像打包

简介：微服务Docker镜像打包视频服务和订单服务

- 视频服务打包
- 订单服务打包

第4集 本地镜像推送阿里云私有镜像仓库

简介：本地多个镜像推送阿里云私有镜像仓库

- 阿里云私有镜像仓库创建
- 网关服务

```
//登录
docker login --username=釉釉cxy registry.cn-shenzhen.aliyuncs.com
//打标签
docker tag 494d49ea5e78 registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-gateway:v1.0
//推送
docker push registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-gateway:v1.0
```

- 订单服务

```
docker tag 06270304d1ec registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-order:v1.0

docker push registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-order:v1.0
```

- 视频服务推送

```
docker tag 8765039e03cd registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
video:v1.0

docker push registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
video:v1.0
```

第十八章 【高级篇】微服务公共组件在阿里云ECS容器化部署

第1集 部署实战-阿里云服务器Docker部署Nacos+镜像加速

简介：阿里云ECS服务器Docker部署Nacos

- 拉取特别慢

```
文档：https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors


路径/etc/docker/daemon.json 增加下面的配置
{
  "registry-mirrors": ["https://pb5bklzr.mirror.aliyuncs.com"]
}

//重启
sudo systemctl daemon-reload
sudo systemctl restart docker
```

- docker拉取镜像

```
docker pull nacos/nacos-server
```

- 查看镜像

```
docker images
```

- 启动Nacos

```
docker run --env MODE=standalone --name
xdclass-nacos -d -p 8848:8848 ef8e53226440 (镜像
id)

//查看日志
docker logs -f
```

- 访问Nacos（记得开放阿里云的网络安全组）

```
http://公网ip:8848/nacos

# 登录密码默认nacos/nacos
```

- 注意docker要启动

```
[root@iZwz91k9y7fpcuu8cqimofZ ~]# docker pull
nacos/nacos-server
Using default tag: latest
Cannot connect to the Docker daemon at
unix:///var/run/docker.sock. Is the docker
daemon running?
```

- 机器配置，nacos安装在高配机器

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服务、订单服务）
```

第2集 部署实战-阿里云ECS服务器Docker部署Sentinel

简介：阿里云ECS+Docker部署Sentinel实战

- docker拉取镜像

```
docker pull bladex/sentinel-dashboard:latest
```

- 查看镜像

```
docker images
```

- 启动Sentinel

```
docker run --name sentinel -d -p 8858:8858  镜像id
```

- 访问Sentinel（记得开放阿里云的网络安全组）

```
http://公网ip:8858

# 登录密码默认sentinel/sentinel
```

- 机器配置, 安装在高配机器(就是微服务同个宿主机)

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服务、订单服务）
```

第3集 部署实战-阿里云ECS服务器Docker部署Zipkin实战

简介：阿里云ECS+Docker部署Zipkin实战

- docker拉取镜像

```
docker pull openzipkin/zipkin:latest
```

- 查看镜像

```
docker images
```

- 启动Zipkin

```
docker run --name xdclass-zipkin -d -p
9411:9411 镜像id
```

- 访问zipkin（记得开放阿里云的网络安全组）

```
http://公网ip:9411/zipkin/
```

- 机器配置, 部署在低配服务器

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服务、订单服务)
```

第十九章 【高级篇】AlibabaCloud微服务阿里云ECS容器化部署

第1集 配置文件Bug修复-网关重新打包推送

简介：脚本bug修复和服务重新打包推送

- 缺少主文件,本地验证 进入target目录 java -jar xxx.jar 执行

```
no main manifest attribute, in /app.jar

xxx.jar中没有主清单属性
```

- 注意事项
  - 重新打包，在聚合工程目录下执行, 不然可能出现类找不到

```
mvn clean
mvn install
```

- 本地退出远程私有镜像仓库

```
docker logout
```

- 重新打包推送到私有镜像仓库（如打包超时，可以多试试几次）

```
            <plugin>

<groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-
plugin</artifactId>
                <executions>
                    <execution>
                        <goals>

<goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <fork>true</fork>

<addResources>true</addResources>
                </configuration>
            </plugin>
```

- 注册中心的ip 改为阿里云局域网ip
- 上述步骤同步更新到网关服务、视频服务、订单服务
- 机器配置, 部署在低配服务器

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服务、订单服务)
```

- 网关打包推送

```
docker tag 046756142a54 registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-gateway:v2.0

docker push registry.cn-shenzhen.aliyuncs.com/xdclass-cloud/cloud-gateway:v2.0
```

第2集 网关配置迁移阿里云Nacos配置中心和服务启动

简介：网关配置迁移和服务启动

- 阿里云Nacos新建配置 dataId = api-gateway-dev.yaml

```
server:
  port: 8888

spring:
  application:
    name: api-gateway
  zipkin:
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比
  cloud:
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
```

```
    gateway:
      routes: #数组形式
        - id: order-service  #路由唯一标识
          #uri: http://127.0.0.1:8000  #想要转发
到的地址

          uri: lb://xdclass-order-service  #从
nocas进行转发

          order: 1 #优先级，数字越小优先级越高
          predicates: #断言 配置哪个路径才转发
            - Path=/order-server/**
            #- Before=2020-09-
11T01:01:01.000+08:00  # 在这个时间点之后不能访问
            #- Query=source  #一定携带这个参数

          filters: #过滤器，请求在传递过程中通过过滤
器修改
            - StripPrefix=1  #去掉第一层前缀

      discovery:
        locator:
          enabled: true  #开启网关拉取nacos的服务
```

- 阿里云网关镜像拉取

```
docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
gateway:v2.0
```

- 网关容器启动

```
docker run  --name xdclass-gateway -d -p
8888:8888 镜像id
```

第3集 视频和订单服务迁移阿里云Nacos配置中心和镜像打包

简介：视频服务和订单服务前移配置中心和镜像打包

- 配置中心新增配置
- 本地镜像打包推送阿里云镜像仓库

```
//视频服务
docker tag 53f5604dcd76 registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
video:v2.0
docker push registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
video:v2.0

//订单服务
docker tag 16e0740b7525 registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
order:v2.0
docker push registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
order:v2.0
```

- 阿里云ecs服务器拉取镜像

```
docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
video:v2.0
docker pull registry.cn-
shenzhen.aliyuncs.com/xdclass-cloud/cloud-
order:v2.0
```

- 机器配置

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服
务、订单服务）
```

- 视频服务配置

```
server:
  port: 9000

spring:
  application:
    name: xdclass-video-service
  zipkin:
```

```
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://127.0.0.1:3306/cloud_video?
useUnicode=true&characterEncoding=utf-
8&useSSL=false
    username: root
    password: xdclass.net

  cloud:
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
    sentinel:
      transport:
        dashboard: 172.18.123.229:8858
        port: 9998

mybatis:
  configuration:
```

```
    log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
```

- 订单服务配置

```
server:
  port: 8001

spring:
  application:
    name: xdclass-order-service
  zipkin:
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比
  cloud:
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
    sentinel:
      transport:
        dashboard: 172.18.123.229:8858
        port: 9999
```

```
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://127.0.0.1:3306/cloud_order?
useUnicode=true&characterEncoding=utf-
8&useSSL=false
    username: root
    password: xdclass.net


# 控制台输出sql、下划线转驼峰
mybatis:
  configuration:
    log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true


#使用随机负载均衡策略
xdclass-video-service:
  ribbon:
    NFLoadBalancerRuleClassName:
com.netflix.loadbalancer.RandomRule

# 开启feign对Sentinel
feign:
  sentinel:
    enabled: true
```

第4集 阿里云ECS服务器快速安装Mysql数据库

简介：阿里云ECS服务器快速安装Mysql数据库

- 安装Mysql 5.7 （注意，Mysql和系统务必保持一致，不然存在不一致）
- 开启mysql远程连接（如果是准线上，建议不要开启远程连接）
- 开放阿里云网络安全组配置 3306 端口

- 导入数据库脚本到Mysql

```
#下载mysql的Yum仓库
wget -i -c http://dev.mysql.com/get/mysql57-
community-release-el7-10.noarch.rpm

yum -y install mysql57-community-release-el7-
10.noarch.rpm

#安装 mysql服务
yum -y install mysql-community-server

#启动数据库服务， systemctl 该命令可用于查看系统状态和管理系统及服务，centos7上开始使用
systemctl start  mysqld.service

#查看状态
systemctl status mysqld.service

#在日志文件中查看初始密码
grep "password" /var/log/mysqld.log

#进入修改Mysql密码

mysql -uroot -p

#新密码设置必须由大小写字母、数字和特殊符号组成
ALTER USER 'root'@'localhost' IDENTIFIED BY
'Xdclass.net168';
```

```
#开启mysql的远程访问，  %是指全部
grant all privileges on *.* to 'root'@'%'
identified by 'Xdclass.net168' with grant
option;

#刷新权限
flush privileges;
```

第5集 AlibabaCloud视频和订单服务阿里云启动

简介：视频服务和订单服务容器启动

- 订单服务新增配置 video.title（配置动态刷新）
- 视频服务和订单服务启动

```
docker run  --name xdclass-video -d -p
9000:9000 53f5604dcd76

docker run  --name xdclass-order -d -p
8001:8001 16e0740b7525
```

- 开放网关服务的网络安全组 8888端口
- 机器

```
120.24.216.117(公)
172.18.123.230（私有）
2 vCPU 1 GiB (安装了Mysql/Zipkin服务)



112.74.55.160(公)
172.18.123.229（私有）
2 vCPU 4 GiB（安装了Nacos、Sentinel、网关、视频服务、订单服务）
```

- 视频服务

```
server:
```

```
  port: 9000

spring:
  application:
    name: xdclass-video-service
  zipkin:
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://172.18.123.230:3306/cloud_video?
useUnicode=true&characterEncoding=utf-
8&useSSL=false
    username: root
    password: Xdclass.net168

  cloud:
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
    sentinel:
      transport:
```

```
        dashboard: 172.18.123.229:8858
        port: 9998

mybatis:
  configuration:
    log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true
```

- 订单服务

```
server:
  port: 8001

video:
  title: alibabacloud

spring:
  application:
    name: xdclass-order-service
  zipkin:
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比
  cloud:
```

```
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
    sentinel:
      transport:
        dashboard: 172.18.123.229:8858
        port: 9999

  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url:
jdbc:mysql://172.18.123.230:3306/cloud_order?
useUnicode=true&characterEncoding=utf-
8&useSSL=false
    username: root
    password: Xdclass.net168


# 控制台输出sql、下划线转驼峰
mybatis:
  configuration:
    log-impl:
org.apache.ibatis.logging.stdout.StdOutImpl
    map-underscore-to-camel-case: true


#使用随机负载均衡策略
xdclass-video-service:
  ribbon:
```

```
    NFLoadBalancerRuleClassName:
com.netflix.loadbalancer.RandomRule

# 开启feign对Sentinel
feign:
  sentinel:
    enabled: true
```

- 网关服务

```
server:
  port: 8888

spring:
  application:
    name: api-gateway
  zipkin:
    base-url: http://172.18.123.230:9411/
#zipkin地址
    discovery-client-enabled: false  #不用开启服
务发现
  sleuth:
    sampler:
      probability: 1.0 #采样百分比
  cloud:
    nacos:
      discovery:
        server-addr: 172.18.123.229:8848
```

```
    gateway:
      routes: #数组形式
        - id: order-service  #路由唯一标识
          #uri: http://127.0.0.1:8000  #想要转发到的地址

          uri: lb://xdclass-order-service  #从nocas进行转发

          order: 1 #优先级，数字越小优先级越高
          predicates: #断言 配置哪个路径才转发
            - Path=/order-server/**
            #- Before=2020-09-11T01:01:01.000+08:00  # 在这个时间点之后不能访问
            #- Query=source  #一定携带这个参数

          filters: #过滤器，请求在传递过程中通过过滤器修改
            - StripPrefix=1  #去掉第一层前缀

      discovery:
        locator:
          enabled: true  #开启网关拉取nacos的服务
```

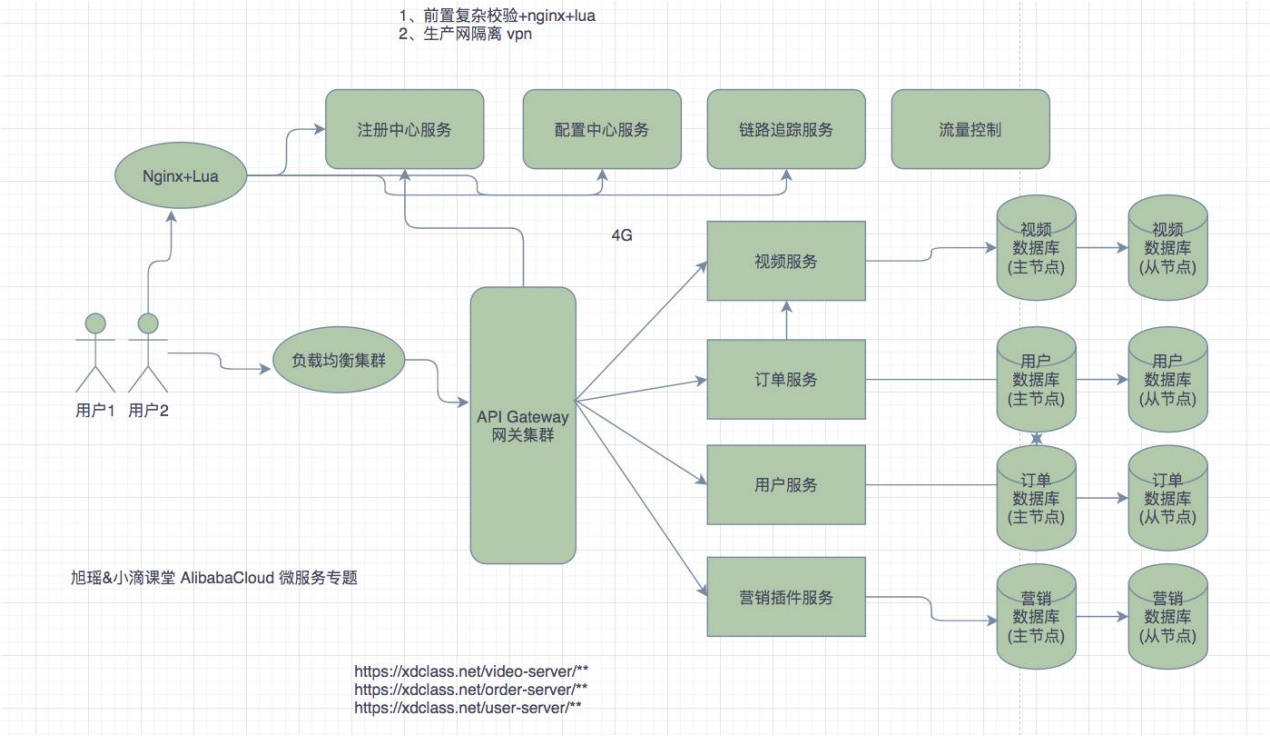
第6集 微服务阿里云部署全链路验证和线上测试

简介：全链路验证AlibabaCloud微服务

- 全链路验证

```
http://112.74.55.160:8888/order-
server/api/v1/video_order/save?videoId=41
```

- 限流验证
- 配置中心动态刷新验证
- 链路追踪验证
- 内外网隔离


1、前置复杂校验+nginx+lua
2、生产网隔离 vpn
注册中心服务
配置中心服务
链路追踪服务
流量控制
Nginx+Lua
4G
视频服务
视频数据库(主节点)
视频数据库(从节点)
用户1
用户2
负载均衡集群
API Gateway网关集群
订单服务
用户数据库(主节点)
用户数据库(从节点)
用户服务
订单数据库(主节点)
订单数据库(从节点)
营销插件服务
营销数据库(主节点)
营销数据库(从节点)
旭瑶&小滴课堂 AlibabaCloud 微服务专题
https://xdclass.net/video-server/**
https://xdclass.net/order-server/**
https://xdclass.net/user-server/**

愿景："让编程不再难学，让技术与生活更加有趣"

第二十章 AlibabaCloud课程总结+高级篇幅规划

第1集 新版AlibabaCloud课程知识点回顾和Github源码地址

简介：课程知识点回顾和课程总结

- 回顾课程知识点
- 学习建议
  - 遇到问题怎么解决?
    - 分析日志和链路，结合搜索引擎
  - 后端项目会一直存在github，地址: https://github.com/jackxy/new-alibaba-cloud-base
  - 不做“有人生，没人养”的教程和项目，如果是我这边的课程的项目，有问题直接问我就可以，也会一直维护下去
- 后端开发人在公司核心工作:

    - 是否需要掌握前端，从开发人员职责来看，前后端分离是趋势。
    - 专业的人做专业的东西，前端工程师负责页面，后端工程师负责接口，效率更高，你是老板怎么考虑?
    - 新的一个方向：全栈工程师，必须把前端和后端方向一起掌握，才能全栈, 但微服务架构的公司很少这样

第2集 从0到高级工程师学习路线规划和高级篇幅介绍

简介：高级工程师路线学习路线规划

- 学习路线
- 推荐社区网站
    - stackoverflow
    - dzone
- 高级篇幅规划
    - AlibabaCloud终极篇
        - 主打源码+原理+高可用架构
    - 综合项目实战
        - 基于现有基础，务必掌握
        - 实战大致选题：秒杀、拼团、网盘、信息流、聚合支付

小滴课堂，愿景：让编程不在难学，让技术与生活更加有趣

相信我们，这个是可以让你学习更加轻松的平台，里面的课程绝对会让你技术不断提升

欢迎加小D讲师的微信： xdclass-lw

我们官方网站：https://xdclass.net

千人IT技术交流QQ群： 718617859

重点来啦：加讲师微信 免费赠送你干货文档大集合，包含前端，后端，测试，大数据，运维主流技术文档（持续更新）

https://mp.weixin.qq.com/s/qYnjcDYGFDQorWmSfE7lpQ

2 全新HTML+CSS零基础入门到进阶网页制作教程

3 全新Javascript零基础多实战例子教程

4 Linux/Centos7零基础入门到高级实战视频教程

5 全新Mysql数据库零基础入门到实战精讲

6 新版Javaweb+jdbc+maven零基础到开发论坛项目实战

7 【面试必备】多线程并发编程+JUC+JDK源码教程

8 新版SSM框架-SpringBoot2.3+Spring5+Mybatis3.x零基础到开发小滴课堂移动端系统综合实战

9 正版Redis4.x零基础整合springboot视频教程

阶段二

中级后端工程师路线

1 IDEA零基础入门到进阶(整合阿里巴巴编码规范)

2 新版Maven3.5+Nexus私服搭建全套核心技术

3 SpringBoot 2.x微信支付在线教育网站项目实战

4 前端-后端必备-Nginx零基础到分布式高并发专题

5 Jenkins持续集成 Git Gitlab Sonar视频教程

6 微服务SpringCloud+Docker入门到高级实战

7 JMeter接口压力测试打造高性能服务

8 Redis高并发高可用集群百万级秒杀实战

9 权限框架Shiro+SpringBoot2.x零基础到高级实战

10 全新JDK8~JDK13全套新特性教程

阶段三

高级后端工程师/技术经理路线

1 Docker实战视频教程入门到高级

2 互联网架构之JAVA虚拟机JVM零基础到高级实战

3 Linux/shell脚本编程视频教程

4 玩转搜索框架ElasticSearch7.x实战