学成在线项目就业指导

1 学成在线是个什么样的项目？

1.1 项目背景

受互联网+概念的催化，当今中国在线教育市场的发展可谓是百花齐放、如火如荼。 按照市场领域细分为：学前教育、K12教育、高等教育、留学教育、职业教育、语言教育、兴趣教育以及综合平台，其中，职业教育和语言教育的市场优势突出。 根据Analysys易观发布的数据显示，预计2019年中国互联网教育市场交易规模将达到3718亿元人民币，未来三年互联网教育市场规模保持高速增长。

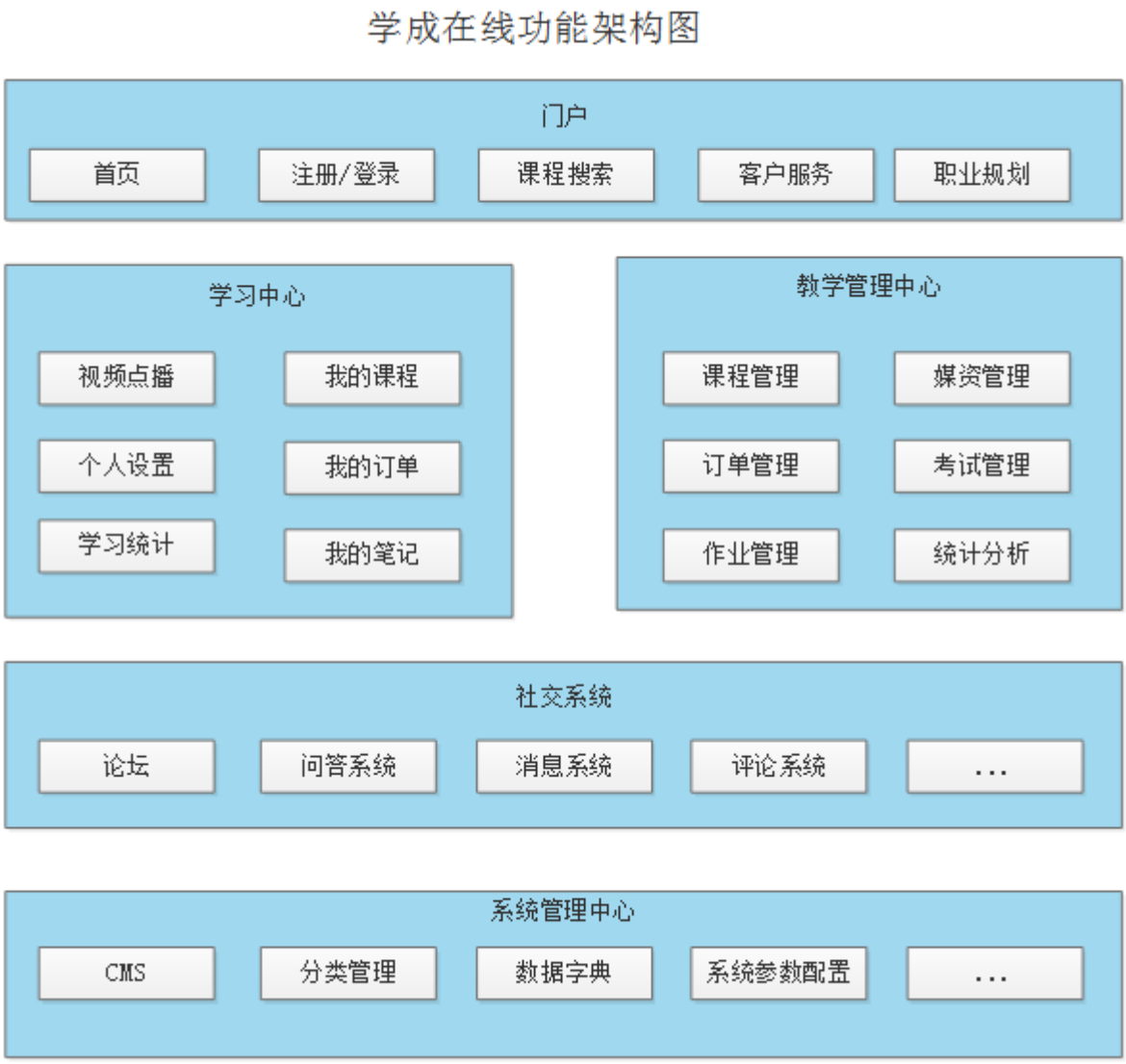
学成在线借鉴了MOOC（大型开放式网络课程，即MOOC（massive open online courses））的设计思想，是一个提供IT职业课程在线学习的平台，它为即将和已经加入IT领域的技术人才提供在线学习服务，用户通过在线学习、在线练习、在线考试等学习内容，最终掌握所学的IT技能，并能在工作中熟练应用。

当前市场的在线教育模式多种多样，包括：B2C、C2C、B2B2C等业务模式，学成在线采用B2B2C业务模式，即向企业或个人提供在线教育平台和学生完成教学活动，市场上类似的平台有：网易云课堂、腾讯课堂等，学成在线的特点是IT职业课程在线教学。

1.2 项目的功能模块

学成在线是一个在线教育平台，提供IT职业课程在线学习，平台包括：门户、学习中心、教学管理中心、系统管理中心、社交系统等子系统。

项目的功能架构如下图：



门户是整个平台的入口，功能包括：门户首页、注册/登录、课程搜索、职业规划，客服等。

学习中心为用户提供在线学习服务，包括：我的课程、视频点播、视频直播、在线考试、在线答疑、学习统计等功能；

教学管理中心为教育机构或个人讲师提供教学管理功能，包括：课程管理、媒资管理、考试管理、问答管理等功能；

系统管理中心提供系统参数配置、CMS、数据字典、分类管理等功能。

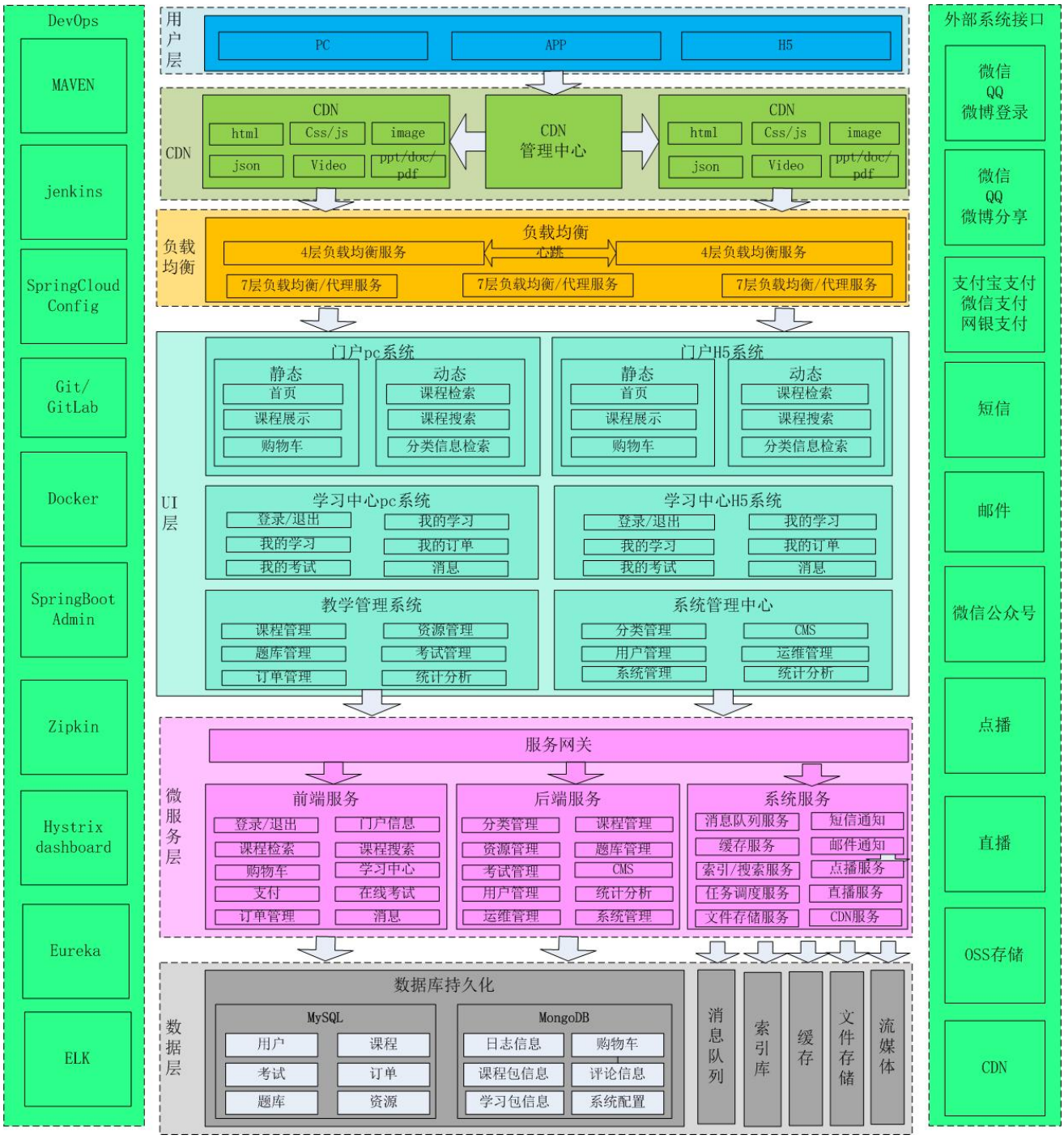
1.3 项目的技术架构

项目采用前后端分离的技术架构，前端采用vue.js技术栈，服务端采用SpringBoot、SpringCloud等Spring全家桶技术栈。

具体见问题2.

2 项目采用什么技术架构？

项目采用前后端分离的技术架构，前端采用vue.js构建，服务端采用Spring Cloud微服务架构，系统分为用户层、CDN、负载均衡、前端UI、微服务层、数据层、接口层及DevOps等部分组成，下图是完整的技术架构图：



业务流程举例：

1、用户可以通过pc、手机等客户端访问系统进行在线学习。

2、 系统应用CDN技术，对一些图片、CSS、视频等资源从CDN调度访问。

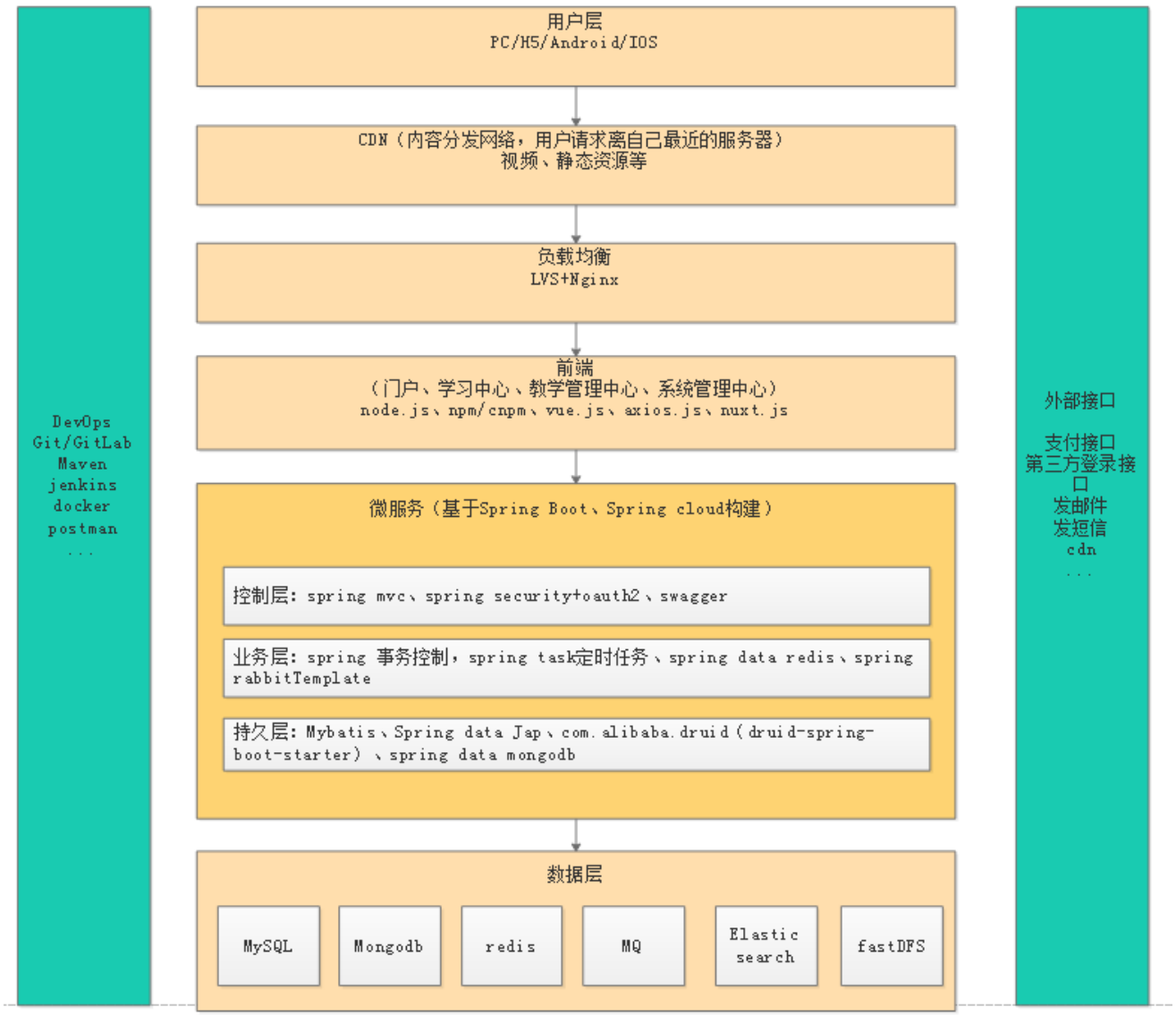
3、所有的请求全部经过负载均衡器。

4、对于PC、H5等客户端请求，首先请求UI层，渲染用户界面。

5、客户端UI请求服务层获取进行具体的业务操作。

6、服务层将数据持久化到数据库。

下图是技术架构简图：



1、用户层

用户层描述了本系统所支持的客户端用户有哪些，本项目目前为各用户提供服务，包括H5、PC、Android和IOS等。

2 CDN全称Content Delivery Network，即内容分发网络，本系统所有静态资源全部通过CDN加速来提高访问速度。系统静态资源包括：html页面、js文件、css文件、image图片、pdf和ppt及doc教学文档、video视频等。

3 负载均衡 系统的CDN层、UI层、服务层及数据层均设置了负载均衡服务，系统采用LVS+Nginx实现负载均衡均衡。

4 UI层 UI层描述了系统向pc用户、app用户、h5用户提供的产品界面。本项目在PC和H5端采用vue.js+elementUI实现。

5 微服务层将系统服务分类三类：前端服务、后端服务及系统服务。 前端服务：主要为学习用户提供学习服务。后端服务：主要为管理用户提供教学管理服务。 系统服务：公共服务，为系统的所有微服务提供公共服务功能。

7 外部系统接口 包括如下接口：

1）第三方登录接口，如QQ、微博、微信等。 2）支付宝、微信支付接口 3）短信接口 （阿里大于）4）邮件接口，通过smpt邮件服务器对外发送电子邮件。 5）微信公众号 6）点播、直播。 7）OSS存储 8）CDN，使用最里云CDN服务加速视频访问速度。

8 DevOps提供了本系统开发、运营、维护支撑的系统，包括如下内容：

Eureka服务治理中心：提供服务治理服务，包括：服务注册、服务获取等。

Docker容器化部署服务：将本系统所有服务采用容器化部署方式。

Maven项目管理工具：提供管理项目所有的Java包依赖、项目工程打包服务。

Git/GitLab代码管理服务:提供git代码管理服务。

Spring Boot Admin服务健康监控：监控微服务的健康状态、会话数量、并发数等。

2.1 微服务技术栈

所有微服务基于Spring Boot、Spring Cloud构建

1）控制层：

Spring MVC、Spring Security Oauth2 、Swagger

2）业务层：

事务控制：Spring

任务处理：Spring Task

数据缓存：Spring Data Redis

消息队列：Spring RabbitTemplate

搜索： Elasticsearch

3) 持久层：

操作MySQL：MyBatis、com.alibaba.druid（采用druid-spring-boot-starter）Spring Data Jpa

操作MongoDB：Spring Data Mongodb

4) 数据层，采用MySQL和MongoDb存储数据，MySQL存储用户、课程等系统核心信息，MongoDb存储

cms、配置信息等。

2.2 接口定义规范

项目架构设立接口层，接口层使用swagger注解描述接口的内容，接口定义规范如下：

1、请求

get 请求时，前端请求key/value串，SpringMVC采用基本数据类型（String、Integer等）或自定义类型接收。

Post请求时，前端请Form表单数据（application/x-www-form-urlencoded）和Json数据(Content-Type=application/json)、多部件类型数据（multipart/form-data），对于Json数据SpringMVC使用@RequestBody注解解析请求的json数据。

2、响应

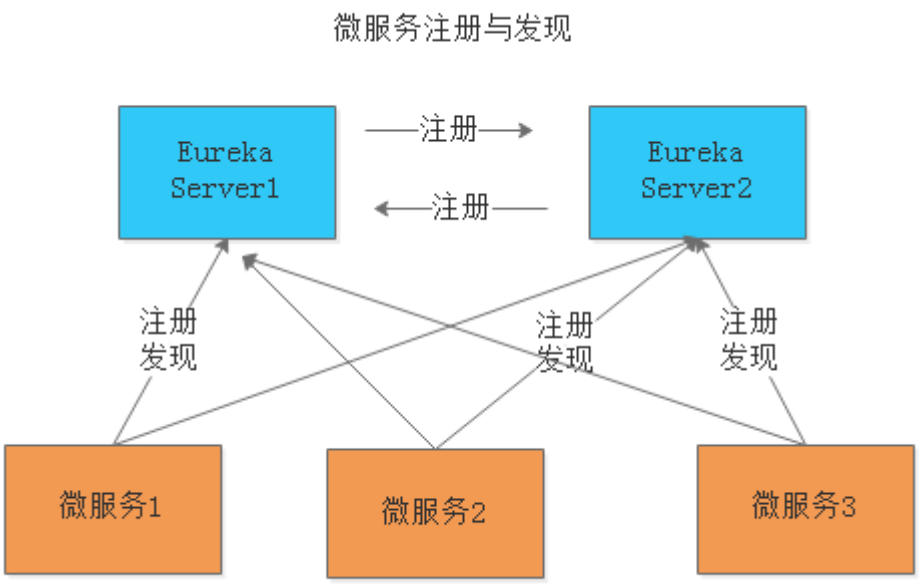
响应结果统一信息为：是否成功、操作代码、提示信息及自定义数据。

响应结果统一格式为json。

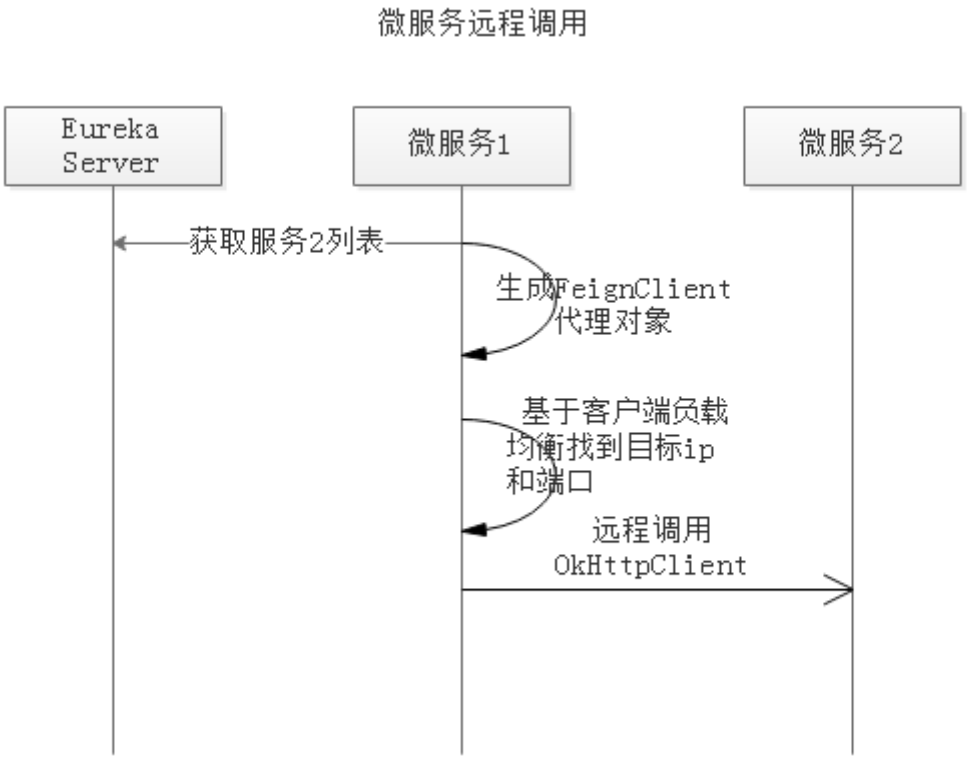
2.3 注册中心

两台Eureka Server互相注册，组成高可用。

微服务向Eureka Server注册自己，并在远程调用时从微服务发现目标服务地址。



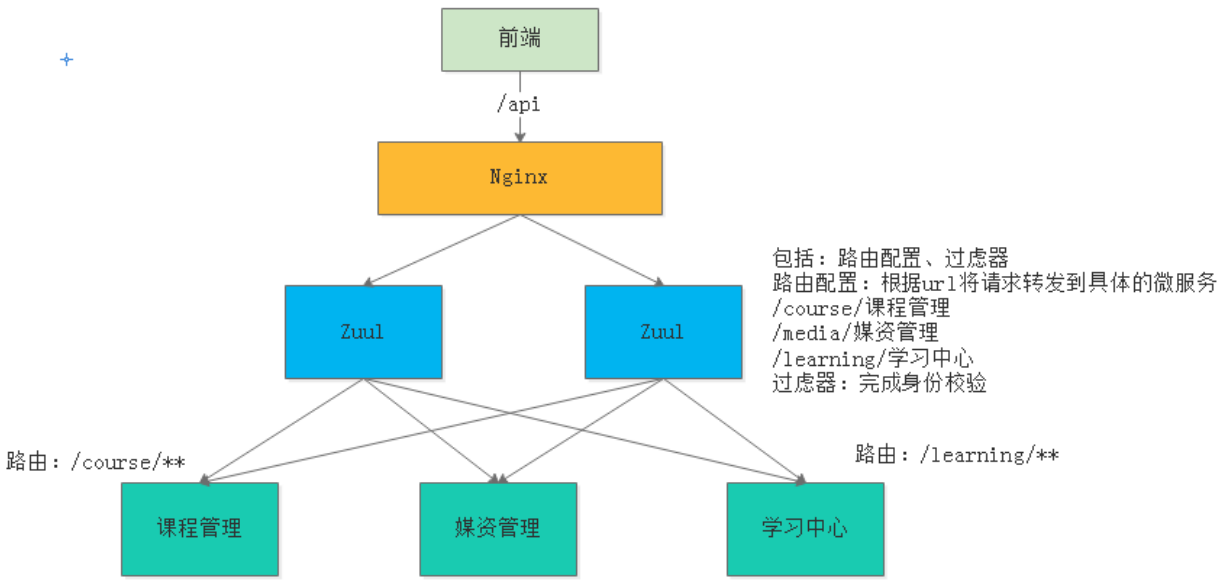
微服务远程调用采用客户端负载均衡技术，使用Feign Client。



2.4 网关

网关的作用是负载均衡、路由转发、请求过虑等。

项目中网关与Nginx配合使用。



2.5 项目使用Spring了吗？用了它的哪些东西？

项目是基于Spring进行构建的:

1、所有的微服务开发采用Spring Boot开发

3、数据层使用Spring Data JPA、Spring Data MongoDB、Spring Data redis。

4、业务层使用Spring来控制本地事务，还使用了Spring Task任务调度框架、Spring AMQP组件等。

5、控制层使用SpringMVC、Sprnig Security Oauth2。

6、微服务管理使用Spring Cloud的Eureka注册中心，微服务之间调用使用Ribbon和Feign Client完成。

7、使用Zuul网关完成微服务安全验证

2.6 Spring Cloud是怎么使用的？

此问题通常是在回答了项目的技术架构后被问及，根据具体的使用Spring Cloud完成微服务开发的步骤来回答即可。

1、每个微服务使用Spring Boot开发，每个微服务工程包括了web、service、dao三层，这和开发一般的项目没有区别：

a、web层使用Spring MVC实现，对外暴露API接口给前端调用。

b、service层就是根据业务逻辑编写JavaBean，并使用Spring的声明式事务控制方式来控制事务。

c、dao层就是数据访问接口，来访问MySQL和Mongodb，访问MySQL使用Spring Data JPA和Mybatis，访问mongodb使用Spring data mongodb。

2、微服务开发完成要向Eureka注册中心注册，以便被其它微服务查找和访问。

3、微服务与微服务之间使用feign来调用，feign Client具有负载均衡的作用。只需要在接口上声明@FeignClient注解，Spring底层会产生动态代理对象，使用ribbon客户端完成调用。

4、前端访问微服务需要通过网关，网关使用Nginx和Zuul来实现，Nginx是最前边的负载均衡，通过Nginx之后便到达了Zuul，项目中Zuul的功能是过虑用户请求，判断用户身份，对于一些对外公开的微服务则需要经过Zuul，直接通过Nginx负载均衡即可访问。

2.7 Spring Data JPA 和 MyBatis为什么两个都用？具体怎么用的？

此问题是考察对数据访问接口的使用程度。

项目中使用Spring Data JPA和MyBatis都是用来访问MySQL，但是它们的分工不同：

Spring Data JPA是Spring 提供的一套JPA接口，使用Spring Data JPA主要完成一些简单的增、删、改、查功能。

对于复杂的查询功能会使用MyBatis编写SQL语言来实现，因为使用Spring Data JPA来做一些复杂的查询是没有MyBatis方便的，Spring Data JPA是面向的对象，而MyBatis直接面向SQL语句。

2.8 什么雪崩？如何解决？

容错保护是指微服务在执行过程中出现错误并从错误中恢复的能力。微服务容错性不好很容易导致雪崩效应，什么是雪崩效应？摘自百度百科中的定义：

收藏 347 6

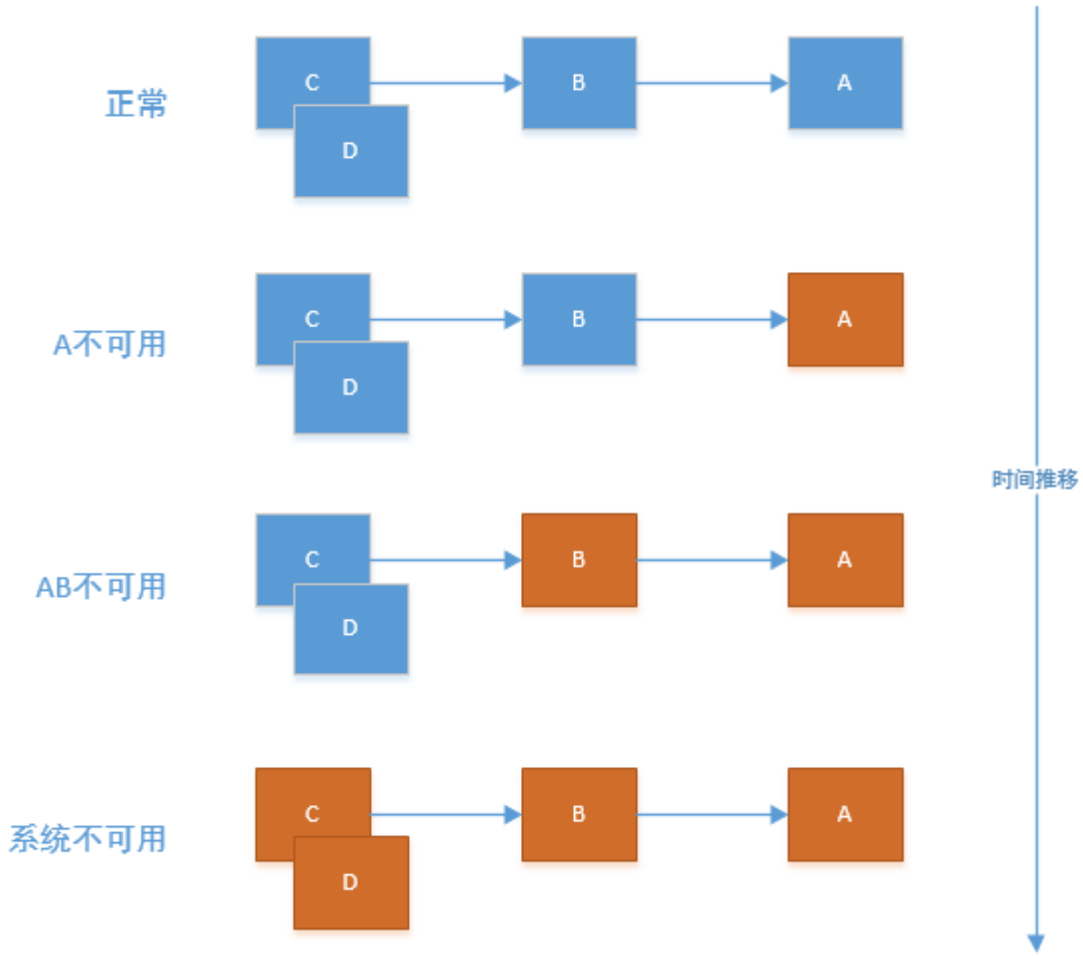
雪崩效应（管理学术语） 编辑

本词条由"科普中国"百科科学词条编写与应用工作项目 审核。

在密码学中，**雪崩效应**（**avalanche effect**）指加密算法（尤其是块密码和加密散列函数）的一种理想属性。雪崩效应是指当输入发生最微小的改变（例如，反转一个二进制位）时，也会导致输出的不可区分性改变（输出中每个二进制位有50%的概率发生反转）。

中文名	雪崩效应	近义词	水滴石穿、蝴蝶效应
外文名	avalanche effect	含　义	一个小的因素导致意想不到的结果

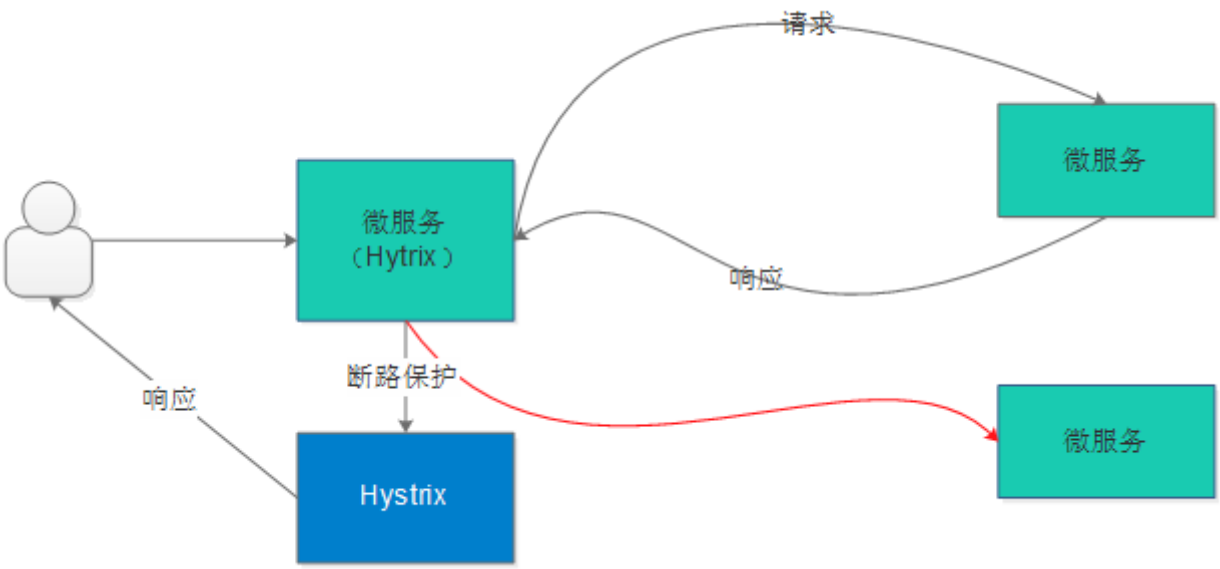
微服务的雪崩效应表现在服务与服务之间调用，当其中一个服务无法提供服务可能导致其它服务也死掉，比如：单点登录服务调用用户信息服务查询用户信息，由于用户信息服务无法提供服务导致单点登录服务一直等待，从而导致用户登录、用户退出功能无法使用，像这样由一个服务所引起的一连串的多个服务无法提供服务即是微服务的雪崩效应。



Spring Cloud Hystrix 是基于Netflix的开源框架Hystrix的整合，它实现了断路保护、线程隔离、信号隔离等容错功能。

断路保护：

断路保护就类似家庭电路中的保险丝，当电路过载时保险丝会自动切断，保护整个电路的安全。微服务的断路保护的工作原理是当请求微服务失败的数量达到一定比例时会切换为开路状态，当请求微服务时就直接返回结果不再请求微服务，当保持开路状态一段时间后判断微服务是否可以正常请求，如果正常则切换到半开路状态，最后切换到哪闭路状态。



具体的操作方法可以采用 Fallback，会每个FeignClient方法调用Fallback，当出现开路则调用Fallback方法返回错误结果。

线程隔离：

调用微服务使用不同的线程池，线程池之间互不影响，即使某个服务不可用也不影响其它服务的调用，比如：对商品服务的调用使用一个线程池，对用户服务的调用使用另一个线程池，即使用户服务不可用也不影响商品服务的调用。

2.9 视图层用什么技术实现？

此问题问的较模糊，没有问是客户端的视图还是服务端的视图，所以此问题不光是视图技术还是考察你对前后端分离的理解。

1、视图层在前端和服务端都存在。

2、前端视图采用vue.js+elementUI产品界面。

3、服务端都是暴露的rest接口，统一用json展示数据。

2.10 接口是怎么定义的？采用什么数据格式？如何实现？

本问题考察前后端分离开发中接口定义技能。

1、接口定义

使用SpringMVC编写Controller方法，对外暴露Http接口，在Controller方法上使用RequestMapping、PostMapping、GetMapping等注解定义Http接口。

2、采用什么数据格式？

分别说明请求和响应：

请求：

get 请求时，前端请求key/value串，SpringMVC采用基本数据类型（String、Integer等）或自定义类型接收。

Post请求时，前端请Form表单数据（application/x-www-form-urlencoded）和Json数据(Content-Type=application/json)、多部件类型数据（multipart/form-data），对于Json数据SpringMVC使用@RequestBody注解解析请求的json数据。

响应：

统一响应json格式。

3、如何实现的？

json格式数据SpringMVC采用FastJson解析为对象。

非json格式数据SpringMVC提供参数绑定的方法，将key/value或Form-Data数据转换为对象或基本数据类型的变量。

3 前后端开发时具体流程是什么？

前后端分离开发模式在互联网公司最常见，特别是一些大型的互联网公司，但是一些传统的软件开发企业仍然是采用传统开发模式，此问题被问及是考察你有没有真正体会前端开发的好处。

1、前端与后端开发人员讨论确定接口。

接口讨论通过，形成接口文档 。

本项目专门设立一个api工程，在此工程定义接口，Spring Boot 集成Swagger，生成Swagger接口，前后端 开发人员通过html查看接口文档的内容。

2、前端与后端开发人员按照接口文档进行开发。

开发过程中各自进行单元测试。

前端人员怎么进行单元测试？

前端人员可以通过一些工具生成一些模拟数据，比如：EasyMock。

有兴趣的同学可自行查阅资料研究。

3、双方功能开发完成进行前后端联调。

阅读：https://github.com/phodal/fe/blob/master/chapters/chapter-13.md

3.1 前端采用什么技术栈？

前端工程大多为单页面应用（SPA），采用vue.js框架开发，搜索功能前端采用nuxt.js服务端渲染（SSR）框架开发。

技术栈包括：

1、node.js

2、vue.js

3、npm/cnpm

4、webpack

5、axios

6、nuxt.js

4 CMS页面管理

CMS （Content Management System）即内容管理系统，本项目对CMS系统的定位是对各各网站（子站点）页面的管理，本项目的CMS系统不去管理每个子网站的全部资源，比如：图片、CSS、html页面等，主要管理由于运营需要而经常变动的页面，从而满足根据运营需要快速开发、上线的需求。

功能包括：

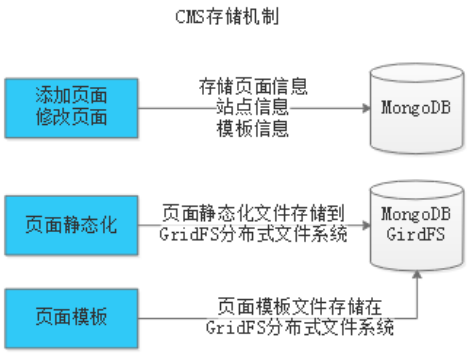
1、站点管理，站点就是本项目各各子网站，站点信息包括：站点名称、站点域名、端口、服务器物理路径等。

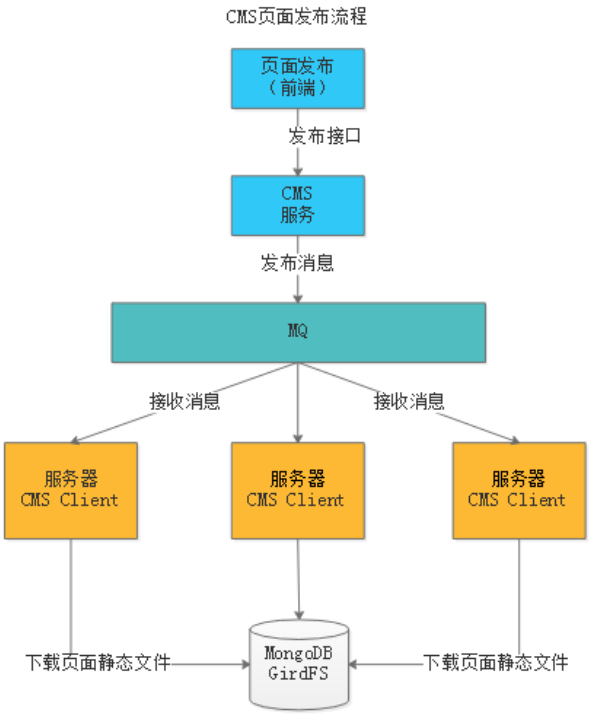
2、模板管理，由于要对页面进行静态化，使用freemarker引擎技术，所以需要定义模板。

3、页面管理，包括：页面添加、页面修改、页面删除等操作。

4、页面预览，对页面静态化，在浏览器预览页面静态化内容。

5、页面发布，将页面静态化后发布到所属站点服务器。





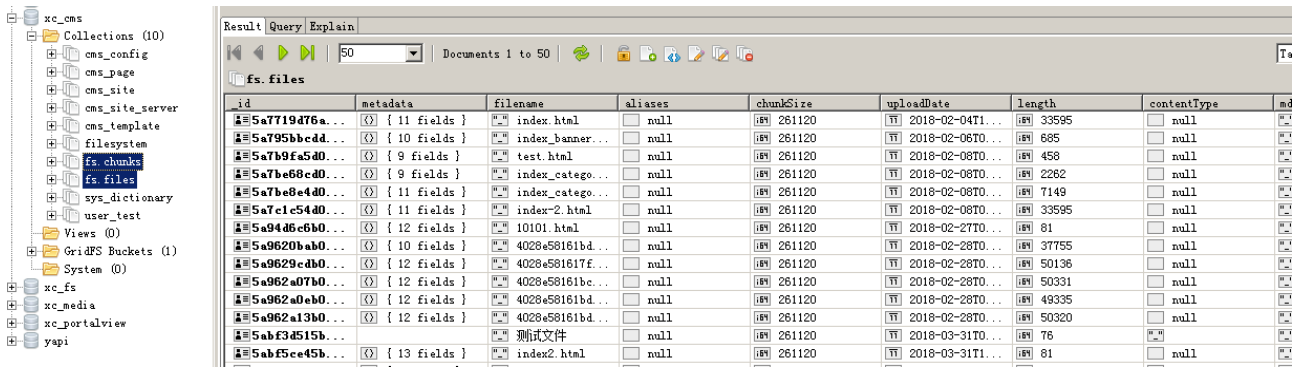
4.1 GirdFS是什么？工作原理是什么？如何使用？

GridFS是MongoDB提供的用于持久化存储文件的模块，它可以作为分布式文件系统使用，CMS子系统将页面文件、模板文件存储到GridFS中，由于本项目使用MongoDB，选用GridFS可以快速集成开发。

它的工作原理是：

在GridFS存储文件是将文件分块存储，文件会按照256KB的大小分割成多个块进行存储，GridFS使用两个集合（collection）存储文件，一个集合是chunks, 用于存储文件的二进制数据；一个集合是files，用于存储文件的元数据信息（文件名称、块大小、上传时间等信息）。

从GridFS中读取文件要对文件的各各块进行组装、合并。

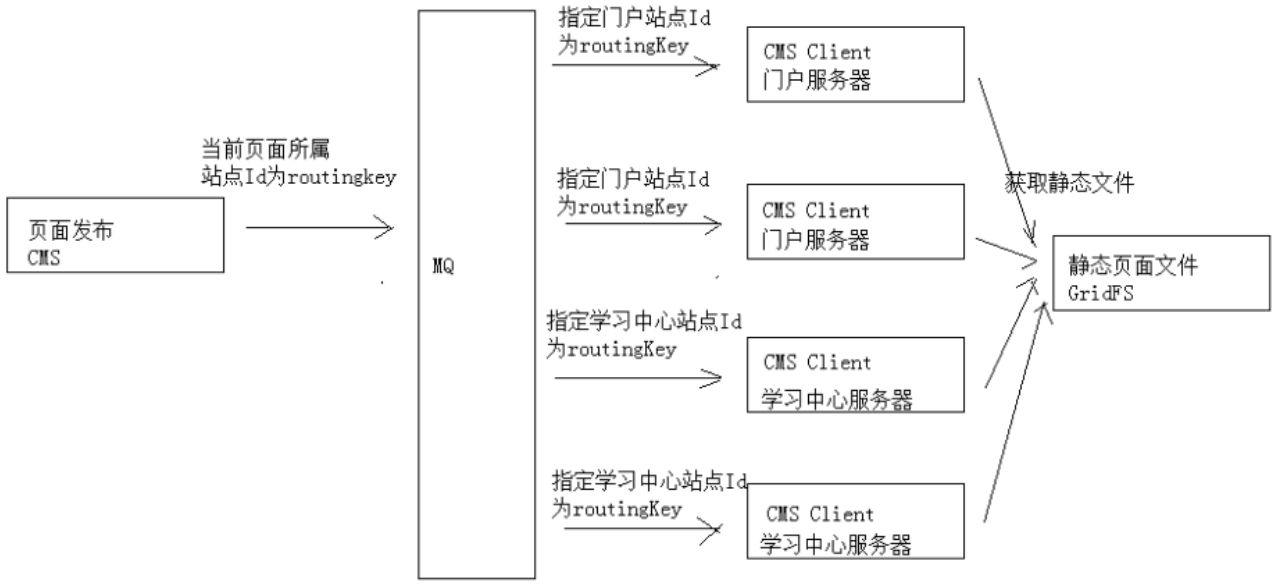


使用方法是：

使用Spring data mongodb包下提供的GridFsTemplate访问GirdFS。

```
gridFsTemplate.findone()      查询文件
gridFsTemplate.delete() 删除文件
gridFsTemplate.store()存储文件
```

4.2 MQ是怎么使用的？



1、平台包括多个站点，页面归属不同的站点，需求是发布一个页面应将该页面发布到所属站点的服务器上。

2、每个站点服务部署CMS Client程序，并与交换机绑定，绑定时指定站点Id为routingKey。

指定站点id为routingKey就可以实现cms client只能接收到所属站点的页面发布消息。

3、页面发布程序向MQ发布消息时指定页面所属站点Id为routingKey，根据routingKey将消息发给指定的CMS Client。

4.3 页面发布的结果如何收集？

每次发布会在数据库记录发布日志，每个CMS Client完成页面发布会上报发布结果。

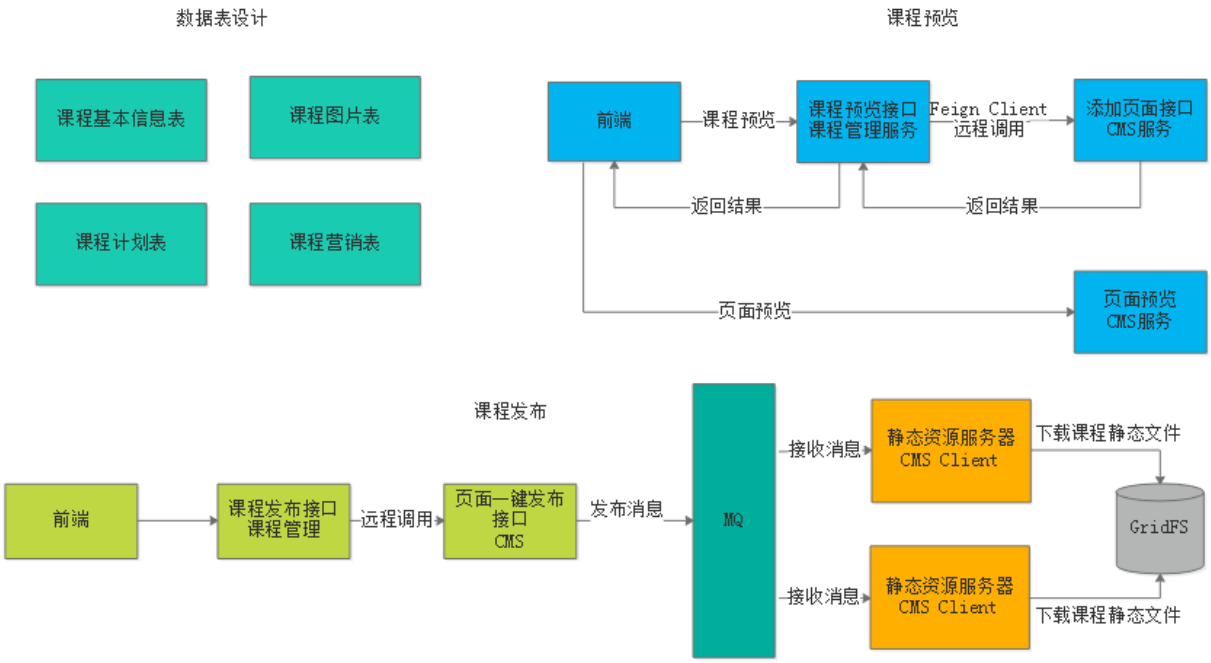
1、在站点管理中配置了每个站点的服务器信息

2、在每次发布页面时会记录发布日志（服务器ID、页面ID、页面名称、发布结果）

3、CMS Client完成页面发布后会向数据上报发布结果。

4、用户通过查询发布日志表的信息就可以知道每一次的发布结果（哪些服务器页面发布成功，哪些发布失败）。

5 课程管理



5.1 为什么用多张表存储课程信息？

1、课程信息比较复杂，为了方便教学机构按步骤管理课程信息，并且也可以划分权限管理课程信息，将课程信息管理功能分为课程基本信息管理、课程图片管理、课程营销信息管、课程计划管理等模块。

2、将课程信息分开也是为了系统扩展需要，如果将课程所有信息存储在一张表中将不利于系统扩展。

5.2 课程图片是如何管理的？

见问题“图片服务器”。

6 媒资管理

每个教学机构都可以在媒资系统管理自己的教学资源，包括：视频、教案等文件。

媒资管理的主要管理对象是课程录播视频，包括：媒资文件的查询、视频上传、视频删除、视频处理等。

媒资查询：教学机构查询自己所拥有的媒体文件。

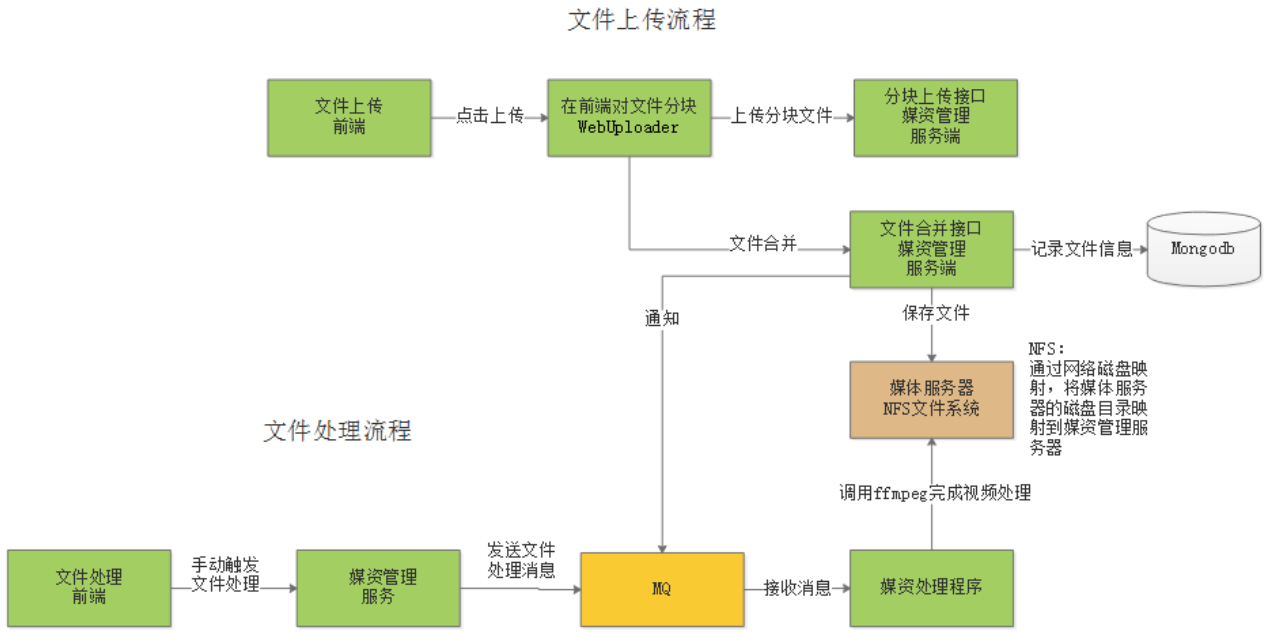
视频上传：将用户线下录制的教学视频上传到媒资系统。

视频处理：视频上传成功，系统自动对视频进行编码处理。

视频删除 ：如果该视频已不再使用，可以从媒资系统删除。

6.1 如何上传大文件？

前端使用WebUploader将文件分块，调用服务端分块上传接口上传分块文件，分块上传完毕前端请求服务端进行合并，当上传过程中断再次进行上传时服务端判断分块是否已经上传，已经上传的分块不再重新上传。



6.2 如何进行视频处理？

上图所示，Java程序调用ffmpeg及流媒体程序员提供的视频处理类库（C程序）完成avi、mp4视频转成m3u8格式的视频。

Java程序使用Jdk提供的Process Builder调用ffmpeg及C程序进行视频处理。

Process Builder可以调用第三方程序，在java程序运行时启动第三方程序进程。

视频处理完成，Java程序捕获第三方程序的输出日志，解析出视频处理完成标记，更新视频处理状态为已完成。

6.3 CDN 内容分发是什么？

视频处理完成会在中心媒体服务器保存一份，另外通过CDN程序将视频发布到边缘媒体服务器，用户点播视频通过CDN请求边缘媒体服务器中的视频，提高了视频播放速度。

具体使用的是第三方公司的CDN服务。

7 搜索

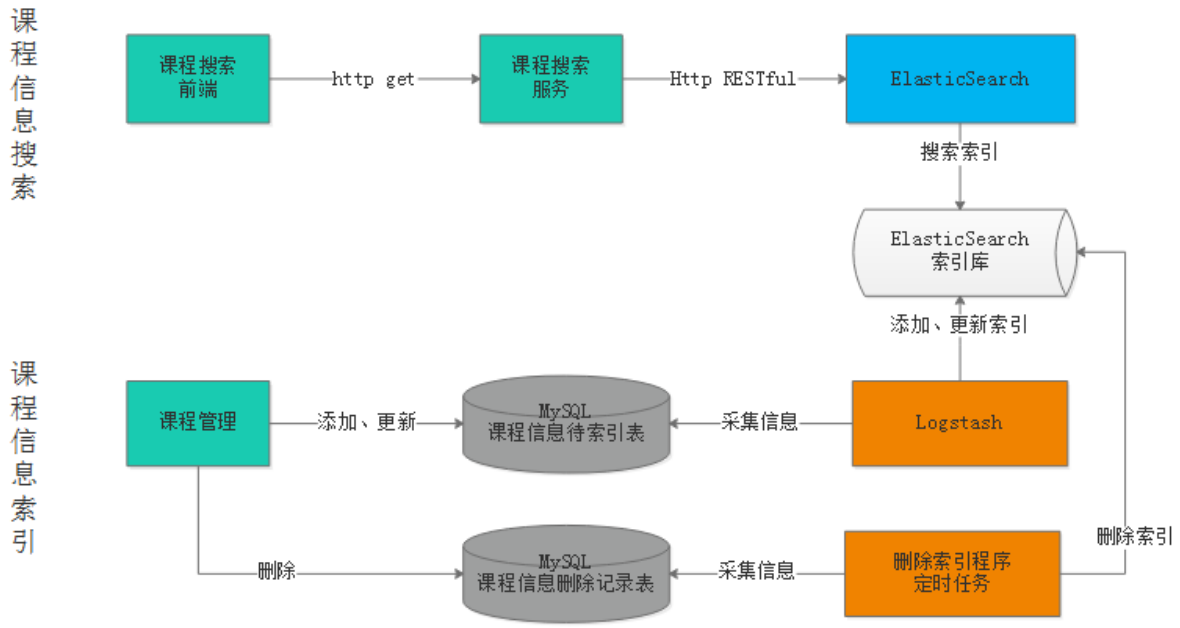
项目中课程搜索采用ElasticSearch来完成。

实现方法是：

1、使用 Logstash（logstash是ES下的一款开源软件，它能够同时 从多个来源采集数据、转换数据）将MySQL中的课程信息读取到ES中创建索引，使用IK分词器进行分词。

2、使用 Java High Level REST Client完成搜索。

3、生产环境使用ES部署为集群。



8 图片服务器

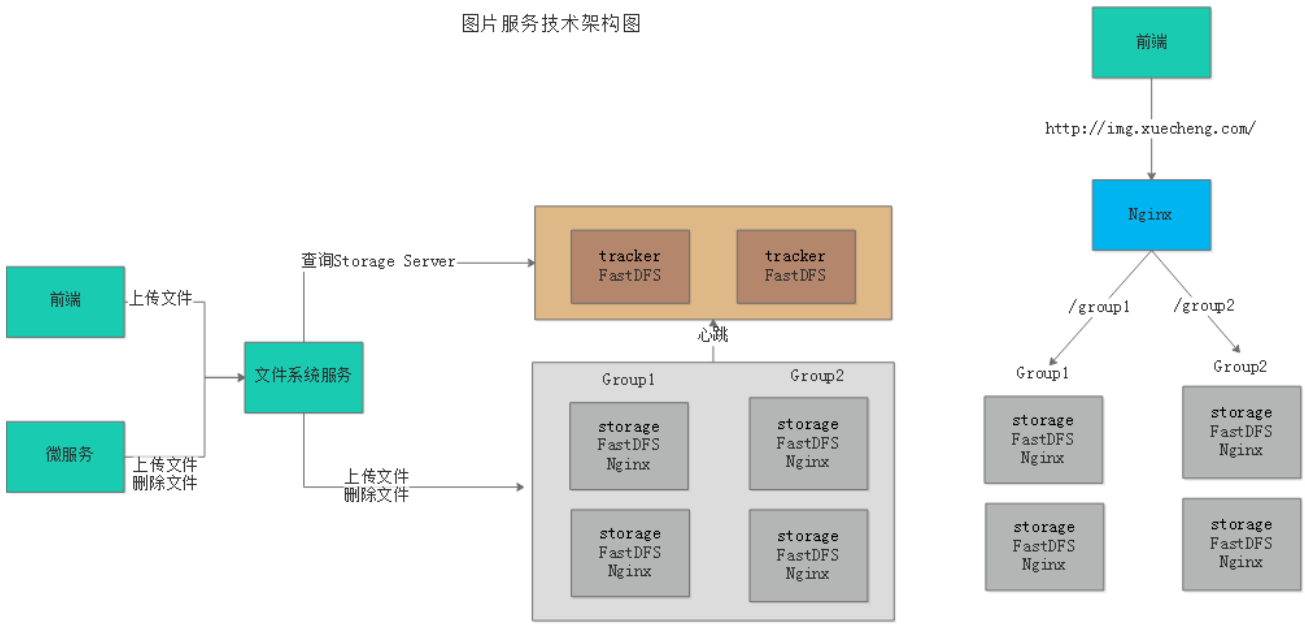
本项目采用fastDFS分布式系统作为图片服务器。

FastDFS是用c语言编写的一款开源的分布式文件系统，适合小文件的存储。

FastDFS包括 Tracker server和Storageserver。客户端请求Tracker server进行文件上传、下载，通过Tracker server调度向Storage server完成文件上传和下载。

使用FastDSF官方提供的Java API实现。

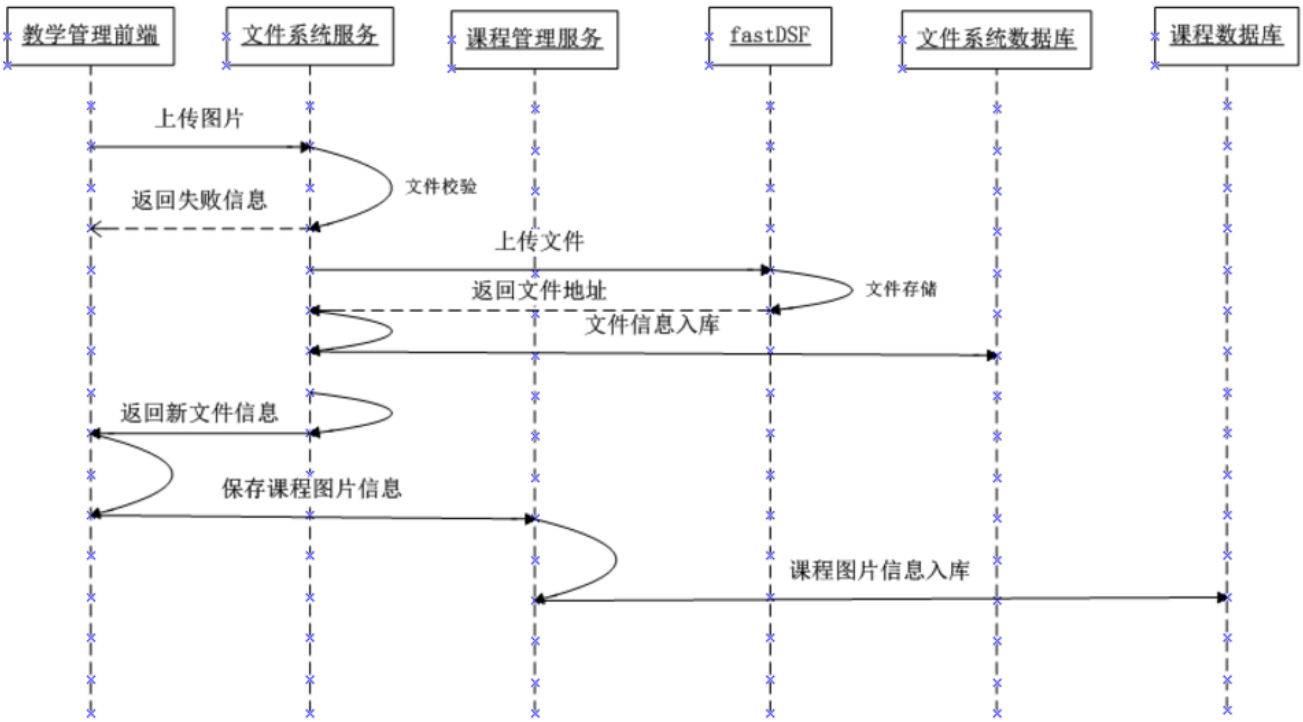
图片服务使用Nginx作为代理服务器，对Storage上部署的Nginx完成负载均衡请求。



8.1 使用FastDFS的好处是什么？

FastDFS相比其它的分布式文件系统它适用小文件存储，它不对文件进行分块存储，也不用对文件进行合并处理，所以性能比GFS、HDFS等通用文件系统的性能要高。

8.2 图片上传流程？



执行流程如下：

1、管理员进入教学管理前端，点击上传图片

2、图片上传至文件系统服务，文件系统请求fastDFS上传文件

3、文件系统将文件入库，存储到文件系统服务数据库中。

4、文件系统服务向前端返回文件上传结果，如果成功则包括文件的Url路径。

5、课程管理前端请求课程管理进行保存课程图片信息到课程数据库。

6、课程管理服务将课程图片保存在课程数据库。

8.3 FastDFS支持断点续传吗？

FastDFS支付断点续传，在Api中有append_file1方法就是用来实现断点续传的，本项目没有使用FastDFS的断点续传功能。

9 认证授权是如何实现的？

本项目采用 Spring security + Oauth2完成用户认证及用户授权。认证授权流程如下：

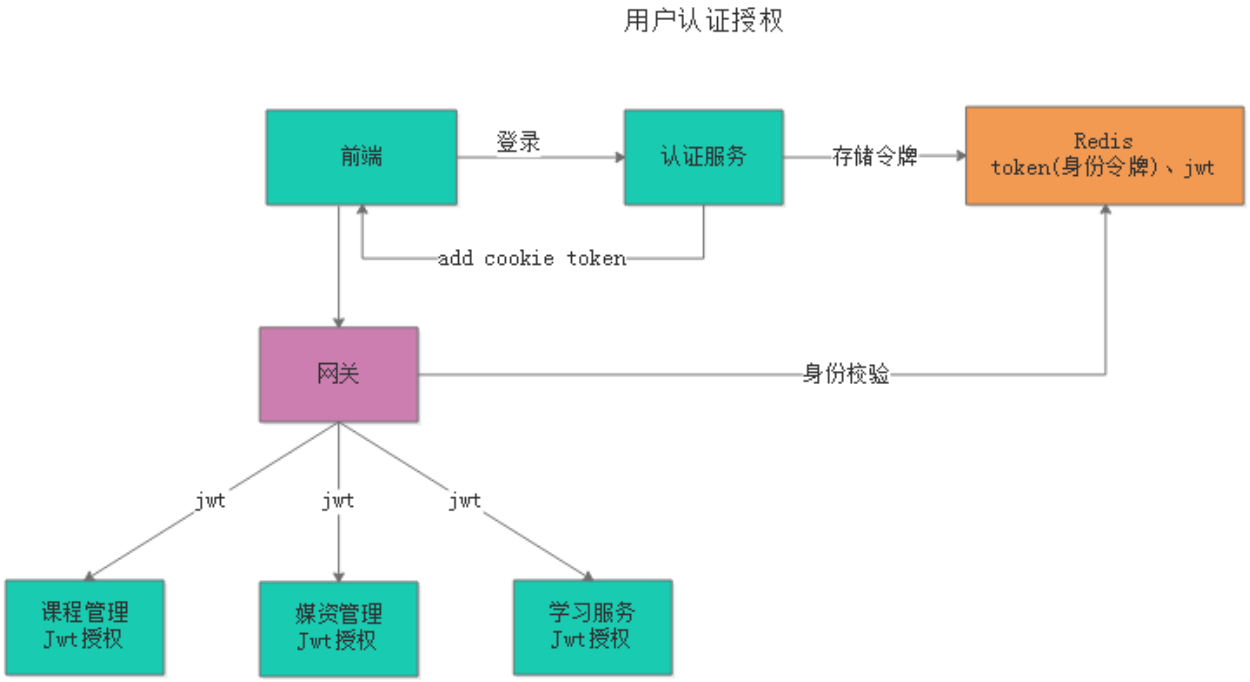
1、用户请求认证服务完成身份认证。

2、认证服务下发用户身份令牌和JWT令牌，拥有身份令牌表示身份合法，Jwt令牌用于完成授权。

3、用户携带jwt令牌请求资源服务。

4、网关校验用户身份令牌的合法，不合法表示用户没有登录，如果合法则放行继续访问。

5、资源服务获取jwt令牌，根据jwt令牌完成授权。



10 事务是怎么控制的？用到分布式事务控制了吗？如何做的？

此问题考察对事务的理解和应用程度。

1、在微服务中使用Spring 声明式事务控制方式进行控制，在Service方法上添加@Transctional注解即可实现事务控制，它控制的是MySQL的本地事务。

2、项目中大量存在分布式事务控制，比如下单支付、课程发布等地址都用到了分布式事务。

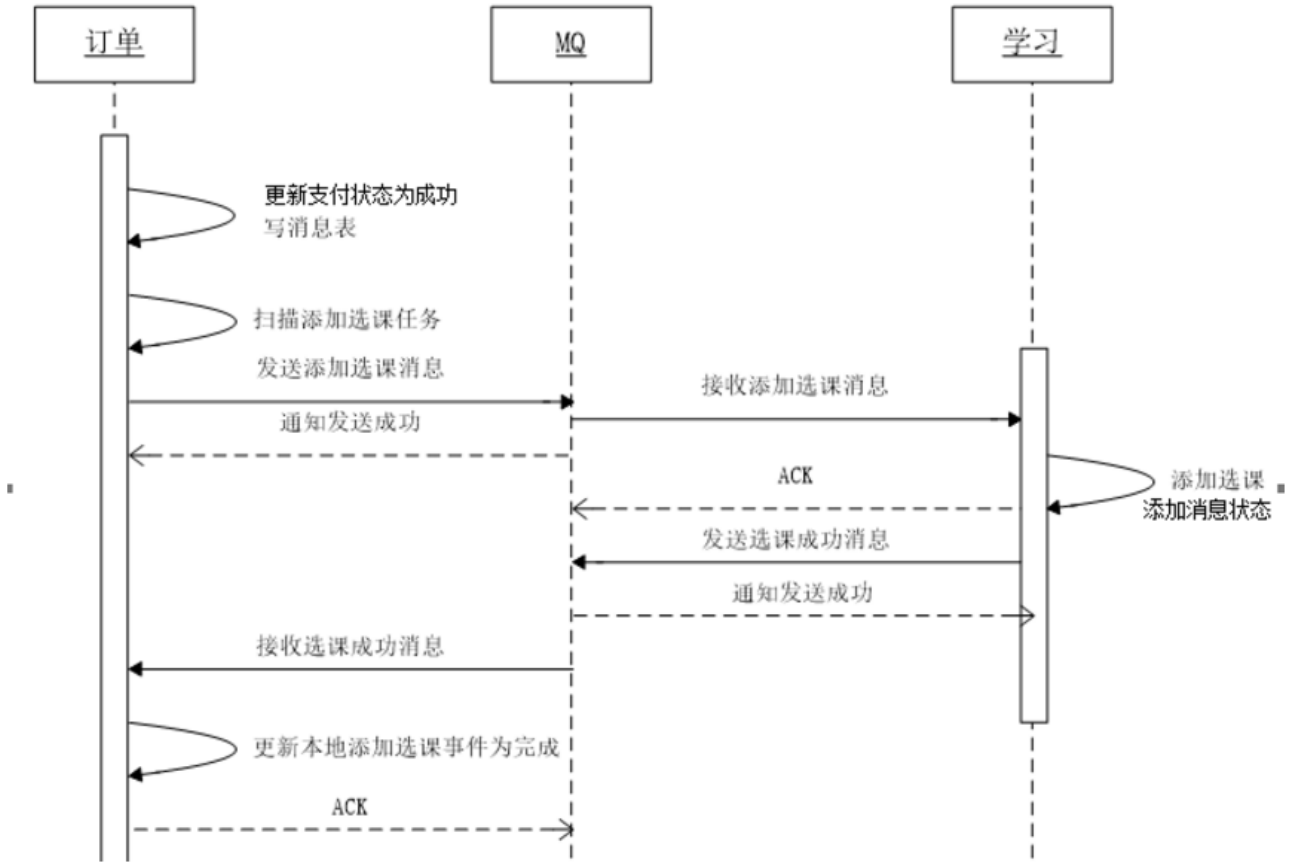
本项目实现分布式事务控制实现最终数据一致性，做法是：

a、将分布式事务拆分为多个本地事务。

b、提交事务前每个参与者要通过数据校验，和资源预留。

c、由消息队列去通知多个事务参与者完成本地事务的提交。

d、提交失败的本地事务会重试。



11 一个接口出现Bug你是怎么调试的？

1、接口的开发需要前端和服务端共同调试，要仔细阅读测试人员反映的bug信息，判断这个bug是服务端的bug还是前端的bug。通常服务接口开发完成会使用postman工具进行测试，测试没有问题再提交到Git或SVN。

2、找到bug的出错点就可以根据bug信息进行修改。

3、修改完成需要前后端再次连调测试，按照测试人员提交的测试流程重新进行测试，测试通过将此bug置为已解决。

12 做过支付接口吗？你是如何做的？遇到什么问题吗？

根据自己的实际情况回答是否做过。

如果做过支付接口则要回答实现过程：

1、系统中收费的课程需要用户在线支付，支付接口采用微信的扫码支付。

2、拿到需求后，确定使用微信支付，首先去阅读微信的接口文档，这里重点阅读统一下单、支付结果通知、支付结果查询三个接口。

3、下载官方提供的sdk编写单元测试用例测试每个接口。测试时没有使用微信的沙箱测试，直接使用正式接口，我们将金额改的小一些进行测试。

4、单元测试通过后开发整个支付功能，最终集成测试通过。

根据自己的实际情况回答开发中遇到的问题：

例子：

接口参数的签名问题，当时是因为自己没有仔细看接口文档导致少写一个必填参数一直报签名失败，随后将所有必填参数填写完成，最终解决问题。

13系统的异常是怎么处理的？

系统对异常的处理使用统一的异常处理流程。

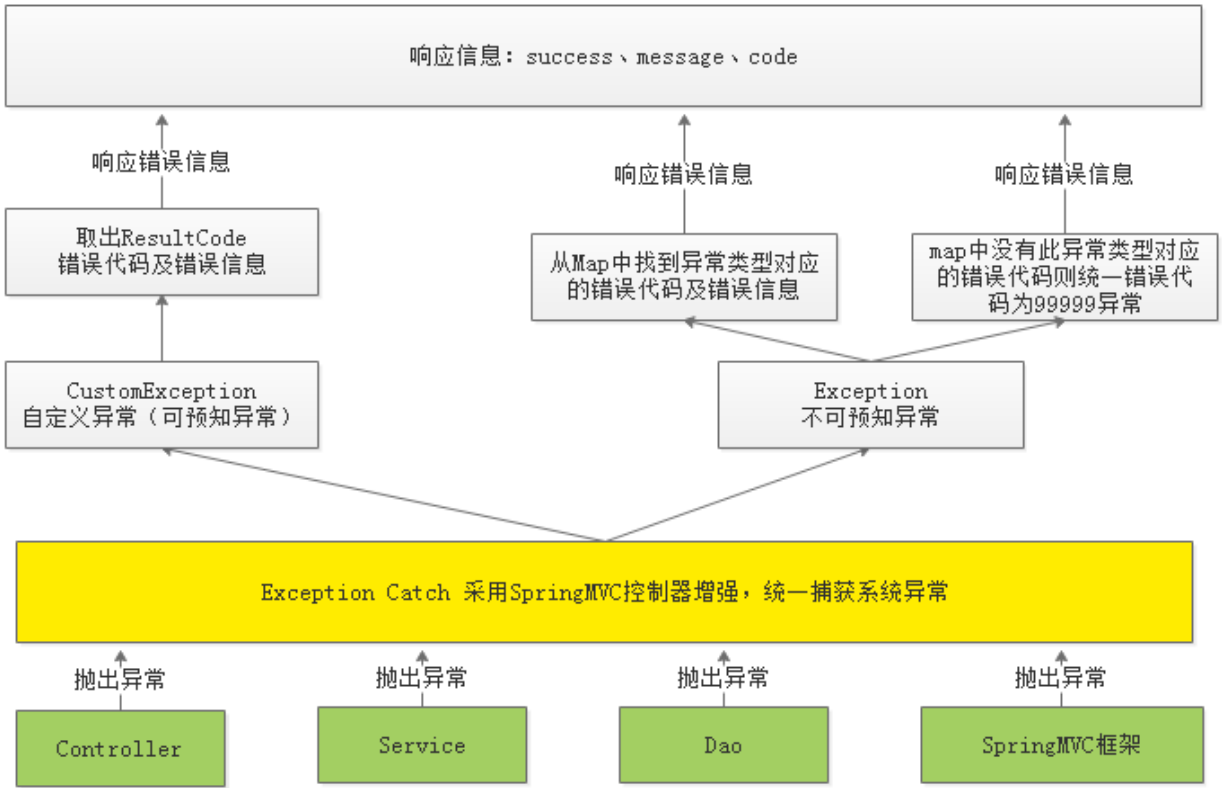
1、自定义异常类型。

2、自定义错误代码及错误信息。

3、对于可预知的异常由程序员在代码中主动抛出自定义异常类型的异常，抛出异常时需要指定错误代码。

4、对于不可预知的异常（运行时异常）由SpringMVC统一捕获Exception类型的异常，由统一的异常捕获类来解析处理，并转换为与自定义异常类型一致的信息格式（错误代码+错误信息）。

5、可预知的异常及不可预知的运行时异常最终会采用统一的信息格式（错误代码+错误信息）来表示，最终也会随请求响应给客户端。



14 使用消息队列了吗？如何使用的？有哪些应用场景？

项目使用RabbitMQ消息队列。

RabbitMQ提供很多的工作模式，如下：

1、Work queues

2、Publish/Subscribe

3、Routing

4、Topics

5、Header

6、RPC

项目主要使用了Routing模式。

Routing模式即路由模式，使用方法是：

1、每个消费者监听自己的队列，并且设置routingkey。

2、生产者将消息发给交换机，由交换机根据routingkey来转发消息到指定的队列。

有哪些应用场景？

1、任务异步处理。

将不需要同步处理的并且耗时长的操作由消息队列通知消息接收方进行异步处理。提高了应用程序的响应时间。

2、应用程序解耦合

MQ相当于一个中介，生产方通过MQ与消费方交互，它将应用程序进行解耦合。

15 视频点播功能是怎么实现的？

本项目采用 HLS 技术实现视频点播。

1、使用FFmpeg对视频进行编码处理，生成m3u8文件及ts文件。

2、使用Nginx作为媒体服务器。

3、客户端使用video.js播放视频。

16 你在开发中遇到什么问题？是怎么解决的？

此问题考察开发人员的问题描述及问题解决能力，可列举开发中实际的技术问题。

回答此问题要从两个方面来回答：

1、问题的描述

2、问题的解决方案

例子：

在处理订单时要用到定时任务，当时采用的是Spring Task来完成，由于一个订单服务会部署多个，多个订单服务同时去处理任务会造成任务被重复处理的情况，如何解决任务的重复处理。

解决：

采用乐观锁解决，在任务表中设置一个version字段记录版本号，取出任务记录同时拿到任务的版本号，执行前对任务进行锁定，具体的做法是执行update根据当前版本号将版本号加1，update成功表示锁定任务成功，即可开始执行任务。