**微服务模块：**项目采用微服务技术架构，根据业务需求，拆分为：内容管理服务、媒资管理服务、搜索服T务、订单支付服务、 学习中心服务、系统管理服务、认证授权服务、网关服务、注册中心服务、配置中心服务、加盟商合作服务、统计分析服务、后台管理系统等。

**技术描述：**

- 使用 Minio 完成媒资系统的大文件上传
- 使用分布式调度组件 XXL-JOB 完成视频处理
- 使用 Nginx 完成静态资源部署、反向代理负载均衡
- 使用 Thymeleaf,实现页面静态化，实现门户首页、课程详情页的页面静态化
- 使用 SpringSecurity，整合 OAuth2.0、JWT 完成分布式统一认证

**项目心得：**

1. 在视频处理过程中因为对视频处理是一个比较复杂的过程，在处理时需要将视频进行分片，因此对于视频的处理就设计到了多任务的处理为了避免一个任务被执行器多次的调用，使用了乐观锁的机制来避免任务的重复的调度。
2. 在课程预览接口中，虽然Thymeleaf的模板机制方便了前端页面生成，但是对于数据不经常变更的网页这样的加载就显的复杂了，因此而且为了提高页面的反应速度，使用了 Thymeleaf提供的页面静态化技术。
3. 在课程发布业务中，为了实现课程信息添同步到加到 redis 以及数据库中，主要是通过 RabbitMq 来实现

内容管理服务、媒资管理服务、搜索服务、认证授权服务微服务核心业务流程的设计与开发

**公司地点：**

**参保人员 19 人**

**人员规模：50 人**

**地点：**山西综改示范区太原学府园区南中环街401号数码港A座B区6层1-3号

老板：毋东明

业务：人工智能基础资源与技术平台；大数据服务；数字技术服务；人工智能基础软件开发。

**一：内容管理服务：**内容管理模块主要对课程及相关内容进行管理，包括：课程的基本信息、课程图片、课程师资信息、课程的授课计划、课程视频、课程文档等内容的管理。内容管理的业务由教学机构人员和平台的运营人员共同完成。

**教学机构人员的业务流程如下：**

1、登录教学机构。

2、维护课程信息，添加一门课程需要编辑课程的基本信息、上传课程图片、课程营销信息、课程计划、上传课程视频、课程师资信息等内容。

3、课程信息编辑完成，通过课程预览确认无误后提交课程审核。

4、待运营人员对课程审核通过后方可进行课程发布。

运营人员的业务流程如下:

1、查询待审核的课程信息。

2、审核课程信息。

3、提交审核结果。

**表：**内容管理模块的基础表涉及9张

**课程查询：**

## 数据模型

1、查询条件：

包括：课程名称、课程审核状态、课程发布状态

课程名称：可以模糊搜索

课程审核状态：未提交、已提交、审核通过、审核未通过

课程发布状态：未发布、已发布、已下线

因为是分页查询所以查询条件中还要包括当前页码、每页显示记录数。

2、查询结果：

查询结果中包括：课程id、课程名称、任务数、创建时间、是否付费、审核状态、类型，操作

任务数：该课程所包含的课程计划数，即课程章节数。

是否付费：课程包括免费、收费两种。

类型：录播、直播。

因为是分页查询所以查询结果中还要包括总记录数、当前页、每页显示记录数。

**接口设计**

**参数设计：**

现在项目中有两类模型类：DTO数据传输对象、PO持久化对象，DTO用于接口层向业务层之间传输数据，PO用于业务层与持久层之间传输数据，有些项目还会设置VO

对象，VO对象用在前端与接口层之间传输数据，

## 创建数据字典表

课程基本信息查询的主要数据来源是课程基本信息表，这里有一个点需要注意，就是课程的审核状态、发布状态。

审核状态在查询条件和查询结果中都存在，审核状态包括：未审核、审核通过、审核未通过三种，下边思考一个问题：一个课程的审核状态如果是“审核未通过”那么在课程基本信息表记录“审核未通过”三个字合适吗？

如果将“审核未通过”五个字记录在课程基本信息表中，显示出来的审核状态就是“审核未通过”这五个字，看起来没有什么问题，如果有一天客户想要将审核未通过的记录在显示时改为“未通过”三个字，怎么办？

这时你可以需要批量处理数据库中记录了，写一个 update 语句，审核状态等于“审核未通过”的全部更新 为“未通过”。看起来解决了问题，如果有一天客户又让改了呢？

和审核状态同类的有好多这样的信息，比如：课程状态、课程类型、用户类型等等，这一类数据有一个共同点就是它有一些分类项，且这些分类项较为固定。针对这些数据，为了提高系统的可扩展性，专门定义数据字典表去维护。

此时我们好像要干 什么了 ，该课程 的审核状态为审核未通过，那么我们在课程基本信息表存储202001，也就是审核未通过对应的代码，这样查询出的数据在前端展示时根据代码取出它对应的内容显示给用户。如果用户要修改“审核未通过”的显示内容只需要在数据字典表修改，无法修改课程基本信息表。

**课程分类查询**：课程分类信息没有在数据字典表中存储，而是由单独一张课程分类表，存储在内容管理数据库中。

这张表是一个树型结构，通过父结点 id 将各元素组成一个树。

**树型表的查询方法**

如果**树的层级固定**可以使用表的自链接去查询，比如：我们只查询两级课程分类，如果**树的层级不确定**，此时可以使用 MySQL递归实现，使用with语法，如下：

```
Java
    WITH [RECURSIVE]
        cte_name [(col_name [, col_name] ...)] AS (subquery)
        [, cte_name [(col_name [, col_name] ...)] AS (subquery)] ...
```

cte_name :公共表达式的名称,可以理解为表名,用来表示as后面跟着的子查询

col_name :公共表达式包含的列名,可以写也可以不写

mysql 为了避免无限递归默认递归次数为 1000，可以通过设置 cte_max_recursion_depth 参数增加递归深度，还可以通过 max_execution_time 限制执行时间，超过此时间也会终止递归操作。

异常处理测试

**二：媒资管理模块：**

主要管理对象是视频、图片、文档等，包括：媒资文件的查询、文件上传、视频处理等。

媒资查询：教学机构查询自己所拥有的媒资信息。

文件上传：包括上传图片、上传文档、上传视频。

视频处理：视频上传成功，系统自动对视频进行编码处理。

文件删除：教学机构删除自己上传的媒资文件。

视频审核：

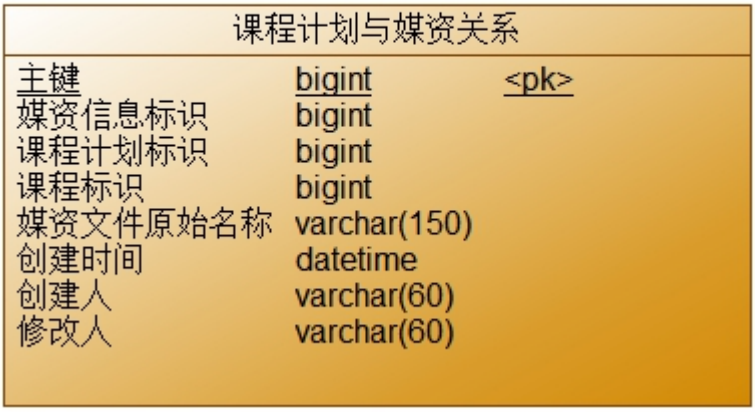
审核媒资包括程序自动审核和人工审核，程序可以通过鉴黄接口(https://www.aliyun.com/product/lvwang?spm=5176.19720258.J_3207526240.51.e93976f4rSq796)审核视频，对有异议的视频由人工进行审核。

**表：**

媒资文件表：存储文件信息，包括图片、视频、文档等。

media_process：待处理视频表。

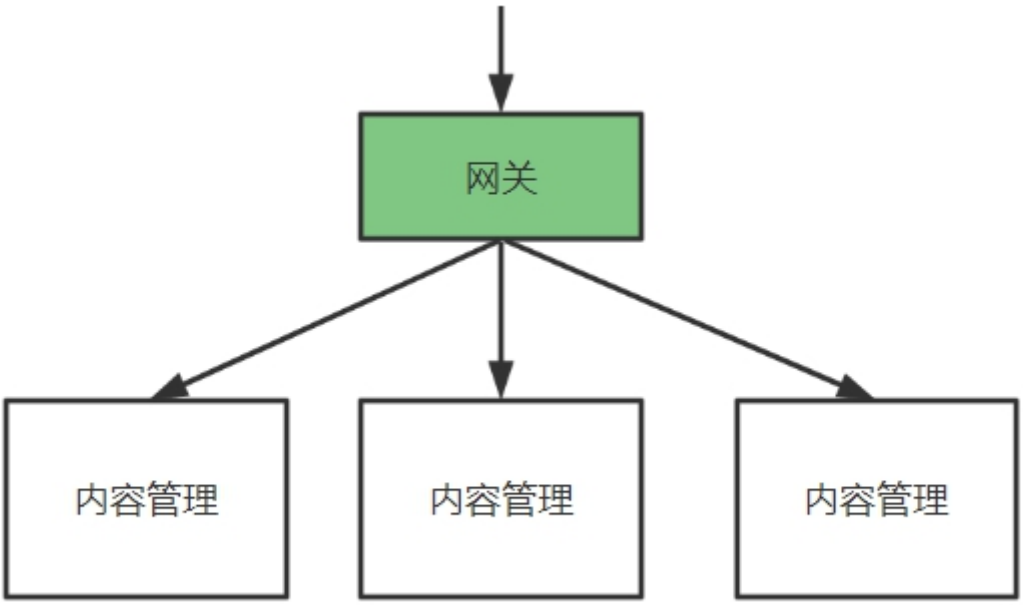
media_process_history：视频处理历史表，记录已经处理成功的视频信息。


网关来同一管理

为什么所有的请求先到网关呢?

有了网关就可以对请求进行路由，路由到具体的微服务，减少外界对接微服务的成本，比如：400 电话，路由的试可以根据请求路径进行路由、根据host地址进行路由等， 当微服务有多个实例时可以通过负载均衡算法进行路由，如下：




另外，网关还可以实现权限控制、限流等功能。

项目采用SpringCloudGateway作为网关，网关在请求路由时需要知道每个微服务实例的地址，项目使用 Nacos作用服务发现中心和配置中心，整体的架构图如下

MinIO集群采用去中心化共享架构，每个结点是对等关系，通过Nginx可对MinIO进行负载均衡访问。

去中心化有什么好处?

在大数据领域，通常的设计理念都是无中心和分布式。Minio分布式模式可以帮助你搭建一个高可用的对象存储服务，你可以使用这些存储设备，而不用考虑其真实物理位置。

它将分布在不同服务器上的多块硬盘组成一个对象存储服务。由于硬盘分布在不同的节点上，分布式Minio避免了单点故障。

Minio使用纠删码技术来保护数据，它是一种恢复丢失和损坏数据的数学算法，它将数据分块冗余的分散存储在各各节点的磁盘上，所有的可用磁盘组成一个集合，上图由 8 块硬盘组成一个集合，当上传一个文件时会通过纠删码算法计算对文件进行分块存储，除了将文件本身分成4个数据块，还会生成4个校验块，数据块和校验块会分散的存储在这8块硬盘上。

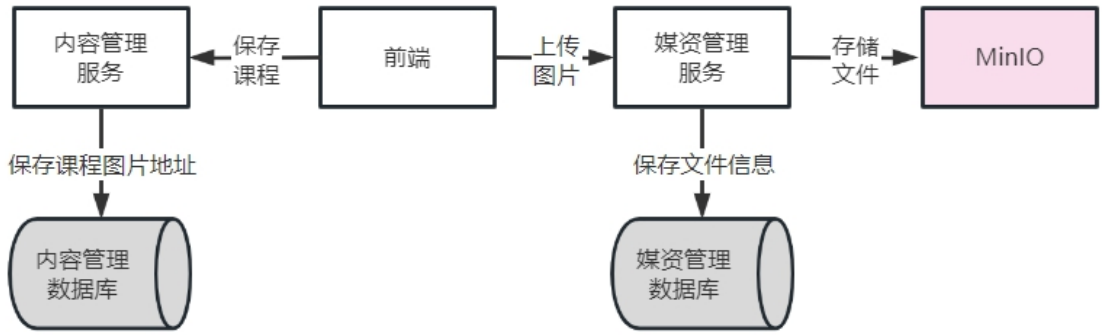
使用纠删码的好处是即便丢失一半数量（N/2）的硬盘，仍然可以恢复数据。 比如上边集合中有4个以内的硬盘损害仍可保证数据恢复，不影响上传和下载，如果多于一半的硬盘坏了则无法恢复。

**文件上传：**

1、上传课程图片前端请求媒资管理服务将文件上传**至分布式文件系统**，并且在媒**资管理数据库保存文件信息。**

2、上传图片成功保存图片地址到**课程基本信息表**中。

详细流程如下:




1、前端进入上传图片界面

2、上传图片，请求媒资管理服务。

3、媒资管理服务将图片文件存储在 MinIO。

4、媒资管理**记录文件信息到数据库**。

5、前端请求内容管理服务保存课程信息，**在内容管理数据库保存图片地址**。

| 媒资信息 | | |
|---|---|---|
| 主键 | bigint | <pk> |
| 机构ID | bigint | |
| 机构名称 | varchar(255) | |
| 文件名称 | varchar(255) | |
| 文件类型 | varchar(12) | |
| 标签 | varchar(120) | |
| 存储目录 | varchar(255) | |
| 文件存储路径 | varchar(512) | |
| 文件id | varchar(120) | <ak> |
| 文件访问地址 | varchar(1024) | |
| 上传人 | varchar(60) | |
| 上传时间 | datetime | |
| 修改时间 | datetime | |
| 状态 | varchar(12) | |
| 备注 | varchar(32) | |
| 审核状态 | varchar(12) | |
| 审核意见 | varchar(255) | |

这里有事务：

@Transactional：

现在的问题其实是一个非事务方法调同类一个事务方法，事务无法控制，这是为什么？

所以判断该方法是否可以事务控制必须保证是通过代理对象调用此方法，

如何解决呢？通过代理对象去调用 addMediaFilesToDb 方法即可解决。

**上传视频：**

：断点续传指的是在下载或上传时，将下载或上传任务（一个文件或一个压缩包）人为的划分为几个部分，每一个部分采用一个线程进行上传或下载，如果碰到网络故障，可以从已经上传或下载的部分开始继续上传下载未完成的部分，而没有必要从头开始上传下载，断点续传可以提高节省操作时间，提高用户体验性。

**流程：**

1、前端对文件进行分块。

2、前端上传分块文件前请求媒资服务检查文件是否存在，如果已经存在则不再上传。

3、如果分块文件不存在则前端开始上传

4、前端请求媒资服务上传分块。

5、媒资服务将分块上传至 MinIO。

6、前端将分块上传完毕请求媒资服务合并分块。

7、媒资服务判断分块上传完成则请求MinIO合并文件。

8、合并完成校验合并后的文件是否完整，如果不完整则删除文件。

使用 minio 合并文件报错：

文件分块的流程如下：

1、获取源文件长度

2、根据设定的分块文件的大小计算出块数

3、从源文件读数据依次向每一个块文件写数据。

文件合并流程：

1、找到要合并的文件并按文件合并的先后进行排序。

2、创建合并文件

3、依次从合并的文件中读取数据向合并文件写入数

# 分布式任务处理、

# XxlJOB：

## 1.调度中心：

负责管理调度信息，按照调度配置发出调度请求，自身不承担业务代码;

**任务执行器：**

负责接收调度请求并执行任务逻辑;

**任务：**负责执行具体的业务处理。

**2.执行流程：**

1.任务执行器根据配置的调度中心的地址，自动注册到调度中心

2.达到任务触发条件，调度中心下发任务

3.执行器基于线程池执行任务，并把执行结果放入内存队列中、把执行日志写入日志文件中

4.执行器消费内存队列中的执行结果，主动上报给调度中心

5.当用户在调度中心查看任务日志，调度中心请求任务执行器，任务执行器读取任务日志文件并返回日志详情

**3.下边思考如何进行分布式任务处理呢?**

执行器在集群部署下调度中心有哪些调度策略呢?

SHARDING_BROADCAST(分片广播)：广播触发对应集群中所有机器执行一次任务，同时系统自动传递分片参数；可根据分片参数开发分片任务；

每个执行器收到调度请求同时接收分片参数。

下边要重点说的是分片广播策略，分片是指是调度中心以执行器为维度进行分片，将集群中的执行器标上序号：0，1，2，3...，广播是指每次调度会向集群中的所有执行器发

送任务调度，请求中携带分片参数。

xxl-job 支持动态扩容执行器集群从而动态增加分片数量，当有任务量增加可以部署更多的执行器到集群中，调度中心会动态修改分片的数量。

每个执行器收到广播任务有两个参数：分片总数、分片序号。每个执行从数据表取任务时可以让任务id模上分片总数，如果等于分片序号则执行此任务。

上边两个执行器实例那么分片总数为 2，序号为 0、1，从任务 1 开始，如下：

1 % 2 = 1 执行器2执行

2 % 2 = 0 执行器 1 执行

3 % 2 = 1 执行器2执行

**4.如果其中一个执行器挂掉，只剩下一个执行器在工作，稍等片刻调用中心发现少了一个执行器将动态调整总分片数为 1。**

**到此作业分片任务调试完成，此时我们可以思考：**

当一次分片广播到来，各执行器如何根据分片参数去分布式执行任务，保证执行器之间执行的任务不重复呢？

通过作业分片方案保证了执行器之间查询到不重复的任务，如果一个执行器在处理一个视频还没有完成，此时调度中心又一次请求调度，为了不重复处理同一个视频该怎么办？

做不到，还需要保证任务处理的幂等性，什么是任务的幂等性？任务的幂等性是指：对于数据的操作不论多少次，操作的结果始终是一致的。在本项目中要实现的是不论多少次任务调度同一个视频只执行一次成功的转码。

1）数据库约束，比如：唯一索引，主键。

2）乐观锁，常用于数据库，更新数据时根据乐观锁状态去更新。

# 课程发布

为了课程内容没有违规信息、课程内容安排合理，在课程发布之前运营方会进行课程审核，审核通过后课程方可发布。

作为课程制作方即教学机构，在课程发布前通过课程预览功能可以看到课程发布后的效果，哪里的课程信息存在问题方便查看，及时修改。

教学机构确认课程内容无误，提交审核，平台运营人员对课程内容审核，审核通过后教学机构人员发布课程成功。

课程发布模块共包括三块功能：

1、课程预览

2、课程审核

3、课程发布

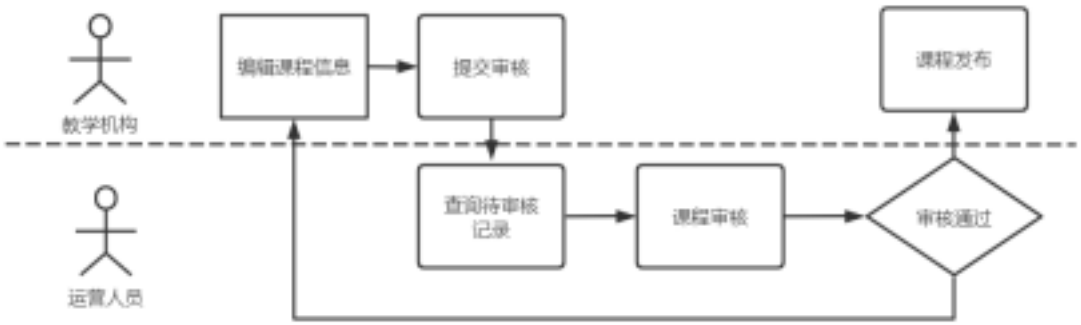
## 课程预览

**教育机构用户**在`课程管理`中可对该机构内所管理的课程进行检索。

## 课程审核

教学机构提交课程审核后，平台运营人员登录运营平台进行课程审核，课程审核包括程序自动审核和人工审核，程序会审核内容的完整性，人员通过课程预览进行审核。

流程：




# 课程预览

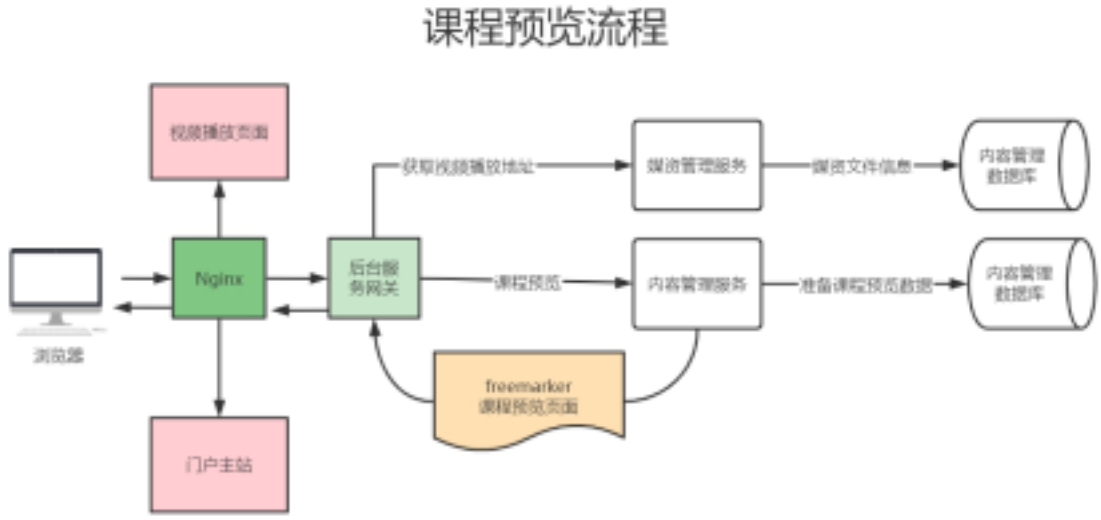
课程预览就是把课程的相关信息进行整合，在课程详情界面进行展示，通过课程预览页面查看信息是否存在问题。

点击课程预览，通过 Nginx、后台服务网关请求内容管理服务进行课程预览。

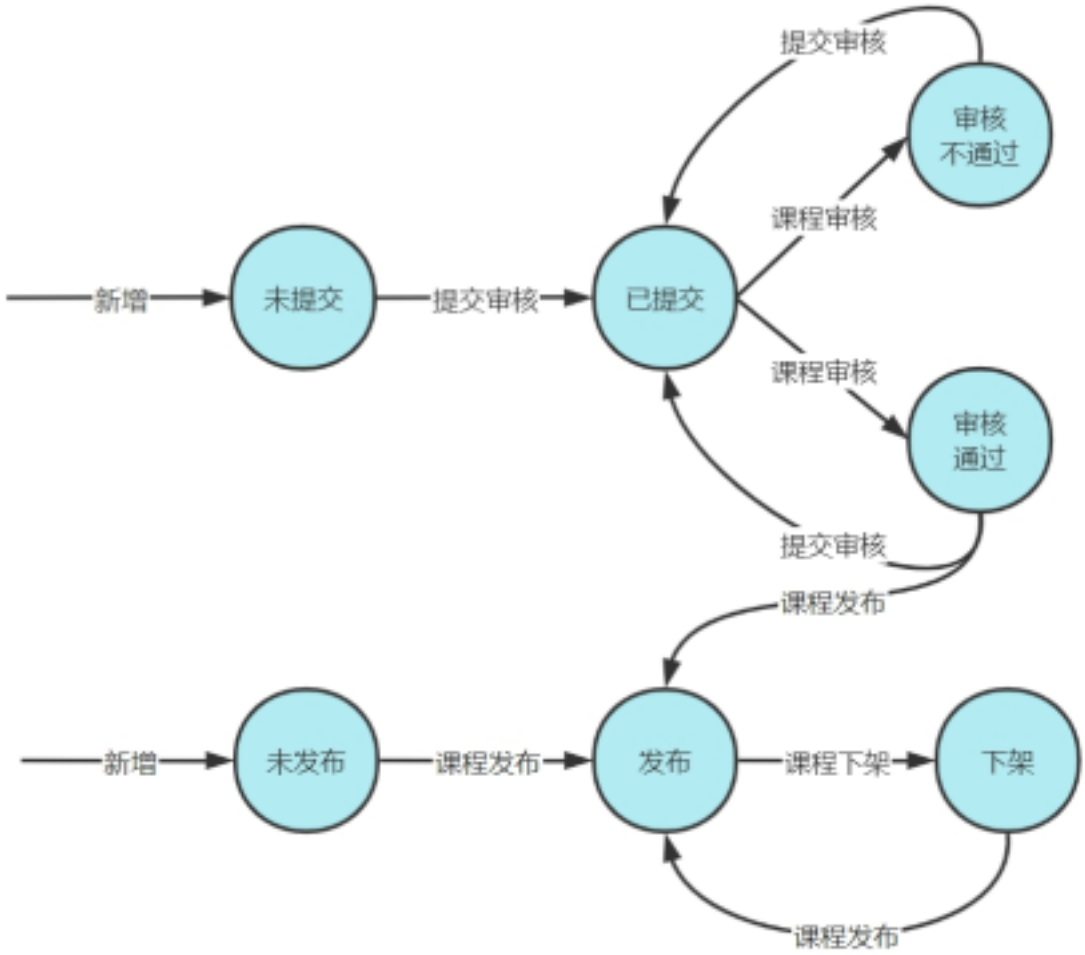
2、内容管理服务查询课程相关信息进行整合，并通过模板引擎技术在服务端渲染生成页面，返回给浏览器。

3、通过课程预览页面点击”马上学习“打开视频播放页面。

4、视频播放页面通过Nginx请求后台服务网关，查询课程信息展示课程计划目录，请求媒资服务查询课程计划绑定的视频文件地址，在线浏览播放视频。




**课程发布：**




1、一门课程新增后它的审核状为”未提交“，发布状态为”未发布“。

2、课程信息编辑完成，教学机构人员执行”提交审核“操作。此时课程的审核状态为”已提交“。

3、当课程状态为已提交时运营平台人员对课程进行审核。

4、运营平台人员审核课程，结果有两个：审核通过、审核不通过。

5、课程审核过后不管状态是通过还是不通过，教学机构可以再次修改课程并提交审核，此时课程状态为”已提交“。此时运营平台人员再次审核课程。

6、课程审核通过，教学机构人员可以发布课程，发布成功后课程的发布状态为”已发布“。

7、课程发布后通过”下架“操作可以更改课程发布状态为”下架“

8、课程下架后通过”上架“操作可以再次发布课程，上架后课程发布状态为“发布”。

问题：

1、课程提交审核后还允许修改课程吗？

如果不允许修改是不合理的，因为提交审核后可以继续做下一个阶段的课程内容，比如添加课程计划，上传课程视频等。

如果允许修改那么课程审核时看到的课程内容从哪里来？如果也从课程基本信息表、课程营销表、课程计划表查询那么存在什么问题呢？

解决：专门设计课程预发布表。

提交课程审核，将课程信息汇总后写入课程预发布表，课程预发布表记录了教学机构在某个时间点要发布的课程信息。

课程审核人员从预发布表查询信息进行审核。

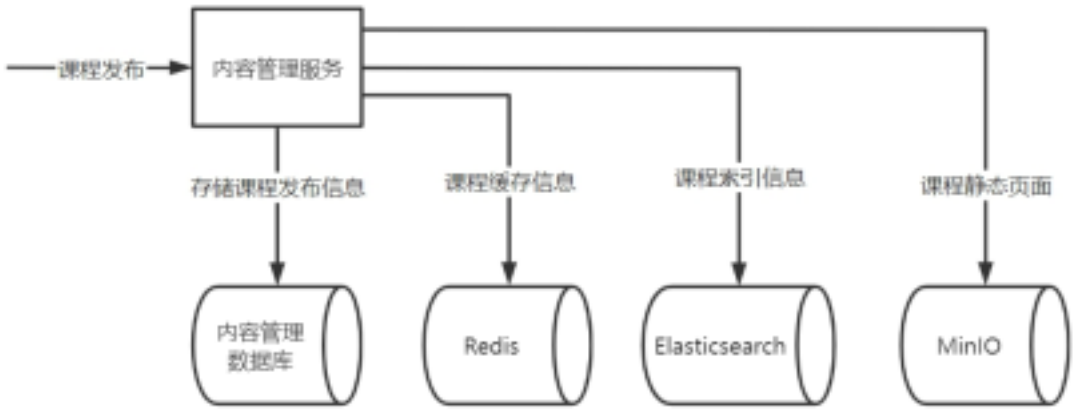
课程审核的同时可以对课程进行修改，修改的内容不会写入课程预发布表。

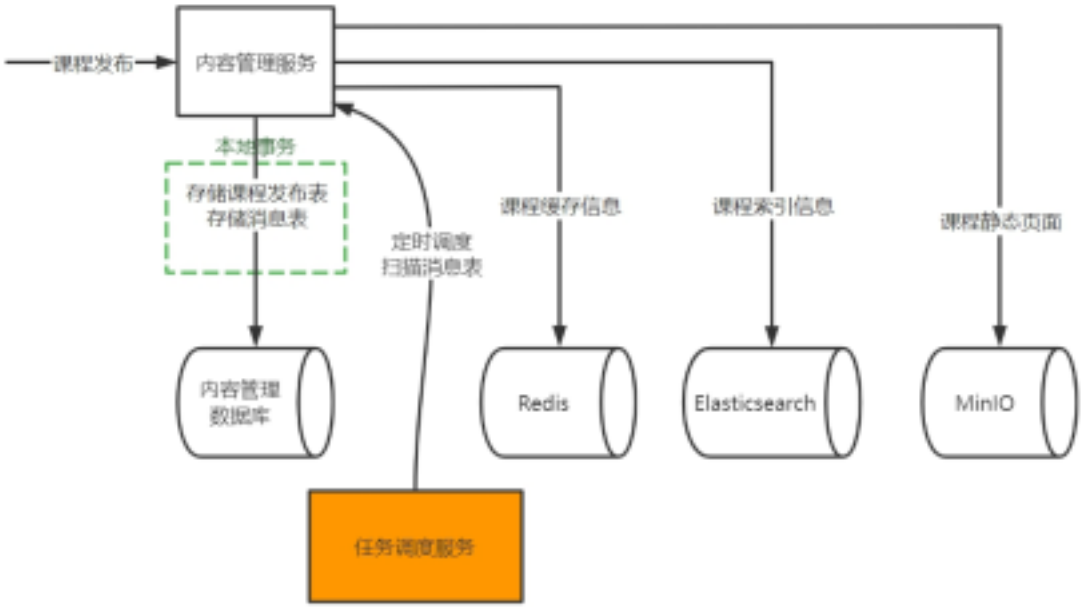
课程审核通过执行课程发布，将课程预发布表的信息写入课程发布表。

2、提交审核课程后，也修改了课程信息，可以再次提交审核吗？

提交审核课程后，必须等到课程审核完成才可以再次提交课程。

**课程发布：**






1、通过一个消息表来实现任务的调度

在内容管理服务的数据库中添加一个消息表，消息表和课程发布表在同一个数据库。

2、点击课程发布通过本地事务向课程发布表写入课程发布信息，同时向消息表写课程发布的消息。通过数据库进行控制，只要课程发布表插入成功消息表也插入成功，消息表的数据就记录了某门课程发布的任务。

3、启动任务调度系统定时调度内容管理服务去定时扫描消息表的记录。

4、当扫描到课程发布的消息时即开始完成向 redis、elasticsearch、MinIO 同步数据的操作。

5、同步数据的任务完成后删除消息表记录。

如何保证任务不重复执行?

采用和视频处理章节一致方案，除了保证任务的幂等性外，任务调度采用分片广播，根据分片参数去获取任务，另外阻塞调度策略为丢弃任务。

注意：这里是信息同步类任务，即使任务重复执行也没有关系，不再使用抢占任务的方式保证任务不重复执行。

还有一个问题，根据消息表记录是否存在或消息表中的任务状态去保证任务的幂等性，如果一个任务有好几个小任务，比如：课程发布任务需要执行三个同步操作：存储课程到 redis、存储课程到索引库，存储课程页面到文件系统。如果其中一个小任务已经完成也不应该去重复执行。这里该如何设计?

# 认证授权

## 单点登录

用户只需要认证一次，便可以在多个拥有访问权限的系统中访问。

spring Security功能的实现主要是由一系列过滤器链相互配合完成。

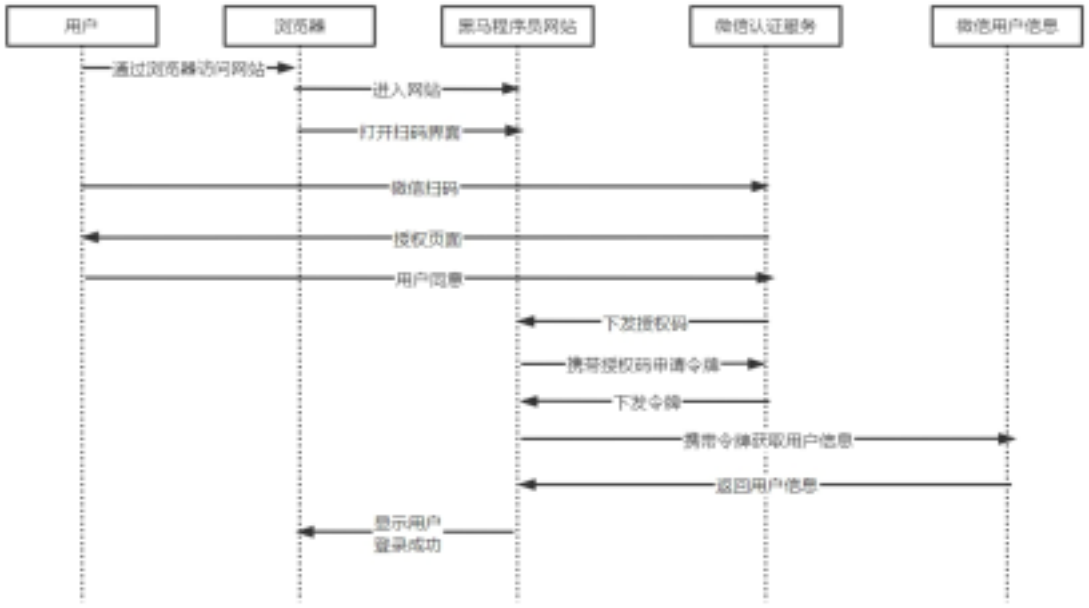

**SecurityContextPersistenceFilter**

**UsernamePasswordAuthenticationFilter** 用于处理来自表单提交的认证。该表单必须提供对应的用户名和密码，其内部还有登录成功或失败后进行处理的 AuthenticationSuccessHandler 和 AuthenticationFailureHandler，这些都可以根据需求做相关改变；

**FilterSecurityInterceptor** 是用于保护 web 资源的，使用 AccessDecisionManager 对当前用户进行授权访问，前面已经详细介绍过了；

**ExceptionTranslationFilter** 能够捕获来自 FilterChain 所有的异常，并进行处理。

## OAuth2




Spring Security 支持 OAuth2 认证，OAuth2 提供授权码模式、密码模式、简化模式、客户端模式等四种授权模式，前边举的微信扫码登录的例子就是基于授权码模式，这四种模式中授权码模式和密码模式应用较多，本节使用 Spring Security演示授权码模式、密码模式，其余两种请自行查阅相关资料。

OAuth2 的几个授权模式是根据不同的应用场景以不同的方式去获取令牌，最终目的是要获取认证服务颁发的令牌，最终通过令牌去获取资源。

AuthorizationServer.java、来配置 OAuth2.0 授权服务器。TokenConfig.java TokenConfig 为令牌策略配置类

授权“XcWebApp”访问自己的受保护资源?

## 测试获取用户身份

jwt令牌中记录了用户身份信息，当客户端携带jwt访问资源服务，资源服务验签通过后将前两部分的内容还原即可取出用户的身份信息，并将用户身份信息放在了

SecurityContextHolder 上下文，SecurityContext 与当前线程进行绑定，方便获取用户身份。

## 网关认证

所有访问微服务的请求都要经过网关，在网关进行用户身份的认证可以将很多非法的请求拦截到微服务以外，这叫做网关认证。

下边需要明确网关的职责：

1、网站白名单维护

针对不用认证的 URL全部放行。

2、校验jwt 的合法性。

除了白名单剩下的就是需要认证的请求，网关需要验证jwt的合法性，jwt合法则说明用户身份合法，否则说明身份不合法则拒绝继续访问。

由于是在网关处进行令牌校验，所以在微服务处不再校验令牌的合法性，修改内容管理服务的 ResouceServerConfig 类，屏蔽 authenticated()。

们要认证服务中连接用户中心数据库查询用户信息。

如何使用 Spring Security 连接数据库认证吗？

前边学习 Spring Security 工作原理时有一张执行流程图，如下图：

用户提交账号和密码由 DaoAuthenticationProvider 调用 UserDetailsService 的 loadUserByUsername()方法获取 UserDetails 用户信息。

我们只要实现 UserDetailsService 接口查询数据库得到用户信息返回 UserDetails 类型的用户信息即可,框架调用 loadUserByUsername()方法拿到用户信息之后是如何执行的，

现在我们修改为 BCryptPasswordEncoder，它是将用户输入的密码编码为 BCrypt 格式与数据库中的密码进行比对。

如何扩展 Spring Security 的用户身份信息呢？

第二也可以扩展 username 的内容 ， 比如存入json 数据内容作为 username 的内容。相比较而言，方案二比较简单还不用破坏 UserDetails 的结构，

、

# 接入微信登录

本项目认证服务需要做哪些事?

1、需要定义接口接收微信下发的授权码。

2、收到授权码调用微信接口申请令牌。

3、申请到令牌调用微信获取用户信息

4、获取用户信息成功将其写入本项目用户中心数据库。

5、最后重定向到浏览器自动登录。

**权限：**

在资源服务集成 Spring Security

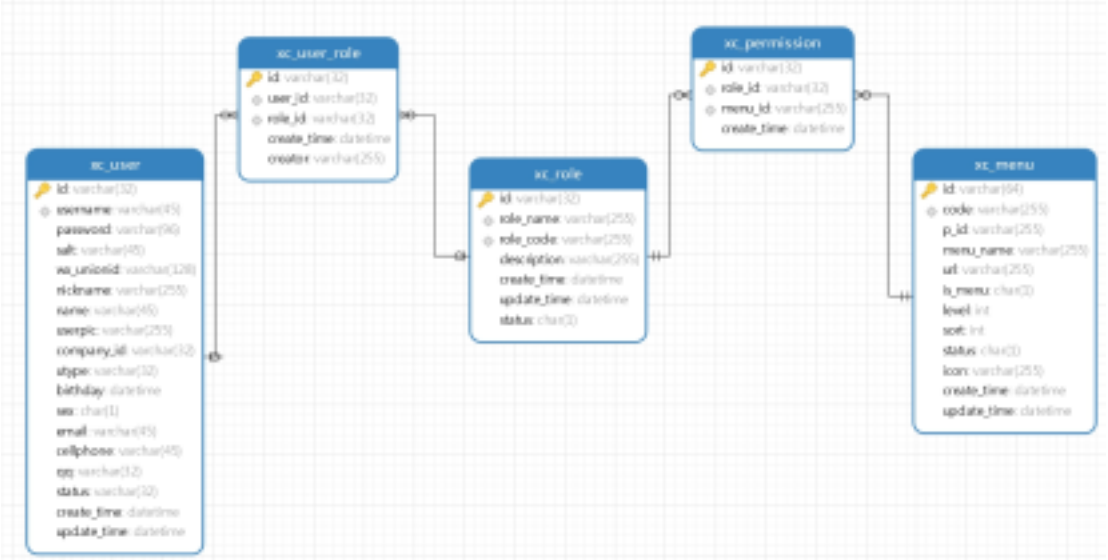
在需要授权的接口处使用@PreAuthorize("hasAuthority('权限标识符')")进行控制

下边代码指定/course/list 接口需要拥有 xc_teachmanager_course_list 权限。

```
@ApiOperation("课程查询接口")
@PreAuthorize("hasAuthority('xc_teachmanager_course_list')")//拥有课程列表查询的权限方可访问
@PostMapping("/course/list")
public PageResult<CourseBase> list(PageParams pageParams, @RequestBody QueryCourseParamsDto queryCourseParams
  return courseBaseInfoService.queryCourseBaseList(pageParams,queryCourseParams);
```

设置了@PreAuthorize表示执行此方法需要授权，如果当前用户请求接口没有权限则抛出异常

org.springframework.security.access.AccessDeniedException: 不允许访问




xc_user：用户表，存储了系统用户信息，用户类型包括：学生、老师、管理员等

xc_role：角色表，存储了系统的角色信息，学生、老师、教学管理员、系统管理员等。

xc_user_role：用户角色表，一个用户可拥有多个角色，一个角色可被多个用户所拥有

xc_menu:模块表，记录了菜单及菜单下的权限

xc_permission:角色权限表，一个角色可拥有多个权限，一个权限可被多个角色所拥有

## 细粒度授权

细粒度授权也叫数据范围授权，即不同的用户所拥有的操作权限相同，但是能够操作的数据范围是不一样的。一个例子：用户A和用户 B都是教学机构，他们都拥有“我的课程”权限，但是两个用户所查询到的数据是不一样的。

根据公司 Id 查询课程，流程如下：

1）教学机构用户登录系统，从用户身份中取出所属机构的 id

在用户表中设计了 company_id 字段存储该用户所属的机构 id.

2）接口层取出当前登录用户的身份，取出机构id

3) 将机构 id 传入 service 方法。

4) service方法将机构id传入Dao方法，最终查询出本机构的课程信息。