Lawrence

实战记录

Spring Cloud前后端分离分布式自媒体平台实战

1. 项目介绍及架构分析
   1. 项目简介

项目基本内容：一个基于Spring Cloud的自媒体运营平台，平台分为三端，作者端、门户端和运营中心端，三端分离分布式设计。

* 1. 微服务架构

1. 访问层：用户访问的客户端，项目前端为web，因此用户可以通过任何设备访问到项目
2. WEB层：包含平台网站、移动端应用和第三方调用；
3. 接口层：包含项目的API网关；
4. 服务层：包含服务通信的中间件，文章管理、用户管理等业务服务模块，日志、注册发现等服务管理模块和为其他模块提供短信、内容安全扫描等的公共基础服务模块；
5. 数据层：即数据存储，包含Mysql数据库、redis缓存、ElasticSearch搜索等技术。

1.3技术架构

1.前端

WEB：Javascript，HTML5，CSS，Vue，Jquery

APP： Android、iOS、小程序等

2.后端

API Gateway：Nginx/Tengine，Zuul

服务层：

负载均衡：Ribbon

熔断降级：Hystrix

服务间通信：RabbitMQ、Rest API、Feign等

注册中心：Eureka、Zookeeper、Consul

数据存储：

缓存：Redis

数据持久化：MySql或MariaDB、MongoDB

文件存储：OSS、FastDFS、GridFS

1. 项目构建
2. 前后端分离开发模式

传统的Java Web开发中，用户的请求通过浏览器，发送到服务器，在服务器进行渲染，如果请求并发量很大，或者需要适配的客户端类型过多，会导致服务器花费大量的处理能力在渲染页面上，导致服务器处理业务的性能下降。而前后端分离的开发模式，来自浏览器的请求会发送到部署在独立容器（Nginx等）上的前端服务器，专注于页面的渲染，后端服务器将专注于处理业务，并且可以使用一套后端服务器代码为不同的前端服务器提供业务服务，降低前后端的耦合性，更容易进行问题分析定位，但在发布过程中会消耗更多的时间和工作精力。

前后端分离模式下，前端与后端使用两套不同的代码，独立部署在两个不同的容器上，其运行互不影响，由两组开发人员维护两套不同的代码库。

1. 前端项目（不包含在开发过程中）

**① 通过Tomcat运行前端项目：**

下载tomcat服务器的程序压缩包，解压，有以下目录：

bin: 包含启动、关闭服务器的脚本文件

conf: 包含相关配置文件

webapps: 存放需要部署的项目app

在bin目录下通过startup和shutdown可以启动或关闭服务器，conf目录下的server.xml是服务器的配置文件，其中Connector标签的port属性指明了服务器部署的端口，默认为8080。

**② 通过Nginx运行前端项目（TBD）**

**To be defined.**

1. 虚拟域名绑定和使用

使用工具：SwitchHosts

通过工具创建新的映射单，将localhost的ip地址127.0.0.1映射到一个因特网地址[www.xxxx.com](http://www.xxxx.com)上，便可以通过该网址加端口号访问网站。

在前端项目代码中，为三端分别都配置了页面的映射路径，在使用前需要确认，映射的端口号与服务器部署的端口号是否一致。

在使用前端项目时，需要注意：app.js中有cookie的使用，需要将cookieDomain改为对应的虚拟域名。

1. 数据库选型和数据导入
   1. 数据库选型：

常用的数据库：

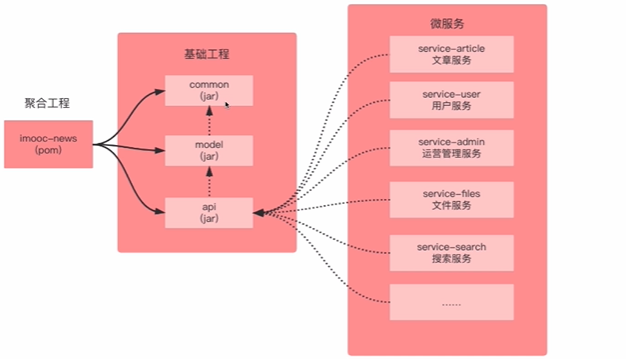
① MySql 常用的数据库，可以直接使用（版本5.6或5.7）

② MariaDB 属于MySql的子项目，与MySql使用相同

* 1. 安装MariaDB（略）
  2. 导入数据库：

使用Mysql命令行，创建新的数据库，通过命令source xxx.sql将转储的sql文件导入该数据库。

1. 构建Maven聚合工程



创建一个Maven构建的工程，作为项目的顶级聚合工程，在配置文件中指定打包方式为pom：

<packaging>pom</packaging>

子工程在这个聚会工程中以module的形式出现，模块之间平级，可以相互依赖。子模块可以使用顶级工程中引入的所有依赖，子模块之间需要使用对方的资源时，需要构建依赖。父工程只负责管理项目使用的依赖和依赖的版本，子工程专注于业务功能的开发。顶级工程的pom文件中，依赖要使用DependenceManager，该标签会自动管理依赖，如果依赖在子工程中没有使用，则不会导入该依赖，直到有模块使用了这个依赖。

1. 创建子工程

右键顶级工程的文件夹，创建两个新的子模块：

① common：枚举、公共方法、基础功能等；

② model：管理所有模型实体类，包含pojo，bean，entity，vo等；

③ api：集中管理api接口（controller）

新建的模块会出现在顶级模块中的pom.xml文件中，这是对子模块的一个定义，在删除模块时，需要删除相应的定义。

因为model会依赖common中的一些基础公共方法，我们需要在model中依赖common。做法与添加外部依赖一致，在pom.xml中添加对common模块的依赖即可。

1. 开发过程中的小技巧：

在完成模块的一定编码后，可以从右侧工具栏找到Maven，点击刚刚修改过的模块，在lifecycle中双击install，对刚刚做过的修改进行模块安装，可以确认在之前的修改中是否有错误。

1. Api接口管理与暴露

API模块中，我们要对接口进行集中管理，并且对外暴露API接口，我们需要在这一模块中引入Spring Boot的相关依赖，创建一个用于存放controller的包，并且为每个模块创建单独的相应的包来存放每个模块的API规范。

为我们需要使用的每个API创建一个interface，在其中为每个方法写好方法原型，并用GetMapping注解指定它们的web映射。回到对应的模块中，让对应的模块依赖API管理模块，并让模块的controller实现我们在另一模块中写好的interface，从而省去实现controller时需要指定的映射路径，只需为Controller指定RestController注解即可。

为什么要这样做？

为Api接口提供向上和向下对接的规范，对于上层，调用者可以通过浏览对应的interface来了解这个接口可以做什么事情，对于下层，开发者可以通过它知道这个接口需要完成什么功能。所有微服务的接口的定义都在这里暴露，实现分布在各自的微服务模块中。这个模块可以管理和查看分散在各个模块中的零散的Controller。此外，微服务的调用都是基于接口的，如果没有这个模块，模块之间的访问会增加相互之间的耦合，通过这个模块可以降低相互访问的模块之间的耦合性。可以提高多服务项目的可拓展性。

另一方面，统一管理的接口更加方便使用Swagger2来进行接口文档化的测试。

1. Logback日志配置与多环境profile
2. Logback日志配置

在模块的resources目录下添加配置文件logback-spring.xml来在当前模块中激活logback的配置，配置文件，关键标签为appender，fileNamePattern和pattern等标签，其中appender标签是定义一个日志输出的关键，它可以是在控制台输出，也可以是生成日志文件。fileNamePattern则是在生成日志文件时，提供文件名生成策略。Pattern则决定了具体日志的格式形式，根据标准的pattern形式，可以自定义各个字段的格式、文字颜色（仅控制台有效）等。

1. 多环境profile

多环境profile是针对多种开发环境，提供不同的配置文件，并在主要配置文件中决定启用哪一个的开发方式。如我们可以在application-dev.yml中指定开发环境使用的配置，并在application.yml也就是主配置文件中指定spring:profiles:active: dev来激活开发环境的配置。

1. 封装返回结果

在Controller中，我们除了返回数据，一般情况会返回更多信息，如http状态码、消息和我们需要返回的数据，用于规范和格式化我们的响应数据。

同时，将http状态码与对应的消息封装在枚举类中，一方面可以让代码更加整洁，另一方面可以提高代码的可维护性和可拓展性。

1. 数据库逆向生成工具

数据库逆向生成工具即将数据库中的数据表逆向生成出后端项目需要使用的实体和实体映射，为mybatis提供数据库数据表到实体类的映射规则。

使用Github开源的mybatis-generator-database来进行实体类的生成，有三点需要注意：

① 访问数据库时，需要禁用SSL连接方式；

② 重复运行generator前，需要清空之前生成的结果，即删除之前生成的mapper.xml文件；

③ 找不到generatorConfig.xml文件时，可以考虑修改generator中的实现为硬编码。

1. 整合mybatis

将生成的代码分别拷贝到各个模块：

① pojo – 拷贝到model模块下

② mapper.xml – 拷贝到对应的业务模块

③ MyMapper.java – mapper接口父类，因为每个模块都用得到，并且每个模块都依赖api模块，故可以放到api模块下（这个操作也许污染了api模块？使得api模块有了另一个更改的缘由，不符合单一职责原则，也许应该放到common包下？）

为业务模块的配置文件添加mybatis的配置，注意各个与包相关的配置必须正确。

课程中此处启动项目会报错，原因是找不到mapper，但是我的项目启动未报错，依据网上的解释，课程提及的相关注解已经被spring boot的启动注解包含了，但是控制台仍然打出了“未找到mapper”的warning。

1. Swagger2 接口文档化调试工具

因为每个业务模块的controller都会依赖于api模块中的接口定义，因此，我们可以只在api模块添加相关配置，从而减少其他模块的配置。在api模块中，先创建swagger2的配置类，注意需要Configuration注解和EnableSwagger2注解。为Swagger2的界面配置一些内容。接下来在controller接口定义中，为接口添加Api注解，让swagger能够扫描到这个接口，可以通过注解属性value和tags为这个接口标注一些信息：

① value：接口标题，在测试时未发现作用

② tags：接口标签，是一个string数组，每增加一个tag，页面会多显示一个接口，但是内容其实是一致的。

随后在方法上方添加注解ApiOperation，同样可以使用value标注标题，notes标注该接口的开发笔记。

随后访问：项目地址:端口/swagger-ui.html即可进入swagger汇总接口的ui界面，使用项目地址:端口/doc.html可以进入文档化界面。

1. 通行证业务开发
2. 短信登陆注册

关键点：

① 短信服务的使用需要做保护，不允许用户在60秒内重复发送验证码。

② 用户接收的验证码存储于缓存中，并设置失效时长。

③ 用户进行一键登录注册时，校验用户是否存在，决定进行登录或注册。

④ 使用过的验证码需要清楚缓存

1. 配置密钥与资源文件
   1. 阿里云-短信服务

阿里云短信服务个人中心→概览→AccessKey，创建一个Access密钥

* 1. 阿里云通用服务配置

在通用业务模块的resources目录下创建新配置文件（xxx.properties），将access key id和access secret配置到该文件下，注意使用驼峰命名法。

在阿里云短信业务的教程中，找到原版SDK，里面的代码中有阿里云需要的依赖，将其复制粘贴到common模块的pom.xml文件中。第三方组件使用的依赖基本都会在common中进行管理，并且版本一般写死。

在模块下新建一个包并在里面新建一个类，用于配置阿里云服务，该类需要Component注解和PropertySource注解，后者的value用于指定该类依赖的properties文件路径，由于我们的文件是在resources目录下的，所以可以直接使用classpath:/xxx.properties，另外，如果我们在properties文件中配置有属性前缀，我们还需要一个ConfigurationProperties注解，将我们的前缀写在注解中的prefix。

在这里需要注意，由于common模块没有Spring Boot相关依赖，注解会访问不到，api中有这些依赖并且api依赖于common，所以这里需要把Spring Boot相关的三个依赖从api模块移动到common模块

至此，阿里云的密钥和id已经通过properties文件和一个类导入到项目之中了。

* 1. 整合短信发送业务

使用阿里云官方提供的代码示例（旧版SDK），直接将main函数内容复制，在Utils包下新建一个SendMessage的工具，将该类标注为Component，并注入我们的resource类。注意此处的QueryParameter不够，需要在左侧填上所有必选项，等待代码更新后，复制所有putQueryParameter方法到代码中。

其中，PhoneNumbers为接收验证码的手机号，我们通过参数传入，SignName为签名名称，是我们自己申请的签名名称，TemplateCode为摸板号，同样来源于申请摸板，TemplateParam为我们给用户发送的验证码，它的值为一个JSON串，因此传参时需要对参数进行处理。