###### 内容回顾

|  |
| --- |
| **1、canal数据同步工具**  （1）把远程库里面内容同步到本地库  **2、SpringCloud组件-Gateway网关**  （1）在客户端和服务端中间一面墙，网关实现请求转发，负载均衡，权限控制等等  *#配置service-edu服务* **spring.cloud.gateway.routes[1].id**=**service-edu spring.cloud.gateway.routes[1].uri**=**lb://service-edu spring.cloud.gateway.routes[1].predicates**= **Path=/eduservice/\*\***  **3、后台系统权限管理模块**  （1）权限管理模块需求  （2）权限管理表和之间关系  \* 五张表：菜单表、角色表、用户表、角色菜单关系表、用户角色关系表  （3）权限管理接口三个基本功能  \* 查询所有菜单  \* 递归删除菜单  \* 角色分配菜单 |

###### Day18内容介绍

|  |
| --- |
| 1、整合Spring Security权限框架  2、整合权限管理前端页面  3、Nacos配置中心  4、提交项目到远程Git仓库 |

###### 整合Spring Security权限框架（框架介绍）

|  |
| --- |
| 1、Spring Security主要包含有两部分：用户认证 和 用户授权  （1）用户认证：  进入用户登录时候，输入用户名和密码，查询数据库，输入用户名和密码是否正确，如果正确的话，认证成功了  （2）用户授权：  登录了系统，登录用户可能是不同的角色，比如现在登录的用户是管理员，管理员操作所有功能，比如登录用户普通用户，操作功能肯定比管理员少很多  2、Spring Security本质上就是过滤器Filter，对请求进行过滤  3、**认证与授权实现思路** |

###### 整合Spring Security权限框架（代码整合）

|  |
| --- |
| **1、创建模块**    **2、在spring\_security模块引入相关依赖**  <**dependencies**>  <**dependency**>  <**groupId**>com.atguigu</**groupId**>  <**artifactId**>common\_utils</**artifactId**>  <**version**>0.0.1-SNAPSHOT</**version**>  </**dependency**>  *<!-- Spring Security依赖 -->* <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-security</**artifactId**>  </**dependency**>  <**dependency**>  <**groupId**>io.jsonwebtoken</**groupId**>  <**artifactId**>jjwt</**artifactId**>  </**dependency**> </**dependencies**>  **3、到源码里面复制整合代码**    **4、复制工具类到common\_utils**      **5、整合代码结构**    **6、在service\_acl模块中引入spring\_security依赖**    **7、创建查询登录和用户权限类** |

###### 权限管理（前端整合）

|  |
| --- |
| **1、到源码中找到整合前端代码，替换项目前端项目内容**    （1）在node\_modules里面替换element-ui依赖  （2）替换其他目录  **2、在项目安装依赖**  **npm install --save vuex-persistedstate**  **3、需要修改两个地方**  （1）修改router文件夹里面index.js里面路径和vue文件地址    （2）修改数据库菜单表路径和页面地址    **4、修改前端项目请求地址是网关地址** |

###### Spring Security代码执行过程

|  |
| --- |
|  |

###### Nacos配置中心(介绍)

|  |
| --- |
|  |

###### Nacos配置中心(读取配置中心配置文件)

|  |
| --- |
| **1、启动Nacos服务**    **2、进入Nacos管理页面，默认用户名和密码nacos**      **3、在Nacos配置中心创建配置文件**  （1）点击 配置列表，点击右边 ＋ 号，进行配置文件创建    **（2）创建配置文件**  **\*\* Data ID 的完整规则格式如下**  **${prefix}-${spring.profile.active}.${file-extension}**  **- prefix** 服务名称  **- spring.profiles.active=dev spring.profiles.active值**  **- file-exetension** 配置文件类型      **4、在项目中读取Nacos配置中心文件**  （1）**springboot配置文件加载顺序**    **（2）在项目中创建bootstrap.properties**    **（3）调用的服务里面引入config依赖**  <**dependencies**>  <**dependency**>  <**groupId**>org.springframework.cloud</**groupId**>  <**artifactId**>spring-cloud-starter-alibaba-nacos-config</**artifactId**>  </**dependency**> </**dependencies**> |

###### Nacos配置中心(读取配置中心配置文件)

|  |
| --- |
| **1、修改项目bootstrap.properties配置文件，添加一行配置**    **2、当配置 文件添加上面那行配置之后，之前创建配置文件读取不到了，需要按照新的规则创建配置文件**    **3、启动项目看到读取新创建配置文件内容** |

###### Nacos配置中心(命名空间切换)

|  |
| --- |
| **1、在Nacos默认命名空间 public，创建三个命名空间**          **2、在不同名称空间创建配置文件**    **使用克隆方式创建**        **3、最终读取**  **（1）修改配置文件，添加命名空间** |

###### Nacos配置中心(加载多个配置文件)

|  |
| --- |
| **1、在dev命名空间创建两个配置文件**    **2、修改项目配置文件，加载Nacos多个配置文件** |

###### Git操作（码云）

|  |
| --- |
| **1、提交过程**    **2、使用Git远程仓库使用码云**  **第一步 进入码云官方网站，注册用户**  [**https://gitee.com/**](https://gitee.com/)  **第二步 创建远程仓库**  **点击右上角的 ＋ 号进行创建**    **填写仓库信息**      **看到创建成功的仓库** |

###### Git仓库（提交代码）

|  |
| --- |
| **1、准备工作：安装git**  （1）在idea配置git环境    **2、创建本地git仓库**  **（1）选择创建本地git仓库**    **（2）选择当前项目目录作为本地git仓库目录**    **（3）创建成功，代码颜色变化**    **3、右键点击项目，选择Git，添加项目代码到本地库**    **（1）选择Git里面Add，添加代码到本地库**    **4、设置远程Git仓库地址**    **（1）设置码云地址**    **（2）点击OK，连接码云远程Git仓库，输入码云用户名和密码**  **\* 第一次需要输入，后面可以保存，之后不需要再输入了**    **5、把本地Git库内容，添加码云远程Git库**                **点击Push就可以了** |

###### Git操作远程库（细节）

|  |
| --- |
| 1、在码云创建仓库时候，不要选择下面内容    2、如果保存输入码云用户名和密码，如何删除？        **3、让本地代码和远程库断开连接**    找到项目文件夹，删除文件夹.git |