链滴

# 工作需要注意的事项

作者：someone9891

原文链接：https://ld246.com/article/1535082691712

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

为了避免不必要的麻烦，标题我改了

今天最后一天上班，整理了一些东西给他们交待一下。后来觉得这些其实日常我们工作中都可能有用所以修改了一下发到社区，截图涉及公司系统，我删掉 ，应该不影响内容。

# 二、部分需要注意的点

## 2.1 权限管理的实现

后台菜单权限管理，是通过登陆以后，从数据库加载权限范围内的菜单，然后修改路由的访问权限来制。
在添加了新路由的同时，需要同步添加到菜单中，才可以在后台进行管理。

## 2.2 验证码

- 目前实现方式

目前实现方式是，在访问登陆界面的时候，先从后台请求一个验证码文本，在前端生成图片，然后登的时候将用户输入的文本 提交到后台进行校验。

验证码请求action地址/user/getVerifyCode

- 存在的问题

目前的实现方式虽然将验证码进行了后台的校验，但是仍然存在安全上的问题，攻击者可以通过先请验证码，再请求登陆接口的方式。

- 可选的改进建议

在访问登陆界面的时候，从后台请求验证码，但是不以文本的形式返回前端去生成图片，而应该直接后台生成干扰的图片后显示再前端，同步将 验证码文本 存储在服务器 ==session== 中，用于 登陆先做服务器验证码校验，校验不通过 则不请求登陆接口。

## 2.3 用户token

- 目前实现方式

在用户登陆以后，判断是否在token，不存在就生成新的==token== ，保存在==redis==中，并返到用户。

生成token 规则 代码路径 com.lezhi.axy.service.utils.Base64Utils,是通过base64编码将用户信息加到token 中，后台代码可以通过解密token获取用户信息，无需从前台传用户信息到后台（因为前台的参数可能会被篡改）。

- 存在两点问题

    - 目前实现时，用户登陆 没有直接生成新的token ，而是先查询是否在redis中存在token，这样的目的是多客户端登陆，就是一个用户名可以在多处登陆，如果每次都生成新的token，会出现 新登的将已经登陆的踢下线的情况（根据业务情况确定一种实现方式即可。）

    - 由于目前是将用户信息加密进token的，如果出现有人==手动在redis中修改了某个用户的toke，或者在数据库修改来用户的信息==，由于上一条登陆时没有生成新的token ，会无法解密出用户

息而导致用户登陆后一直报错。*解决方式可以是删除redis中该用户的token记录，重新登陆生成新即可，或者考虑每次登陆重新生成token（参考上一条说法）*

## 2.4 数据权限

● 由于在起初开发各个接口的时候没有考虑数据权限的问题进去，目前只能通过修改接口的方式添加据权限，但是这种方式存在 ==工作量大、难以维护、容易出错== 的问题。

● 常用的数据权限控制方案：

通过 利用Spring的AOP思想，进行数据的统一拦截。拦截在 getaway 层或者service 层均可进行。果在getaway层进行，就是通过统一判断，向service 添加参数 即可，但需要 getaway 中 请求 servic 时统一的参数格式（比如统一使用WhereParams,这样在 AOP拦截以后，可以做到 将参数添加Wher Parmas中，然后各个接口再添加自己的参数）

但是建议是将 数据过滤做在service 层，通过在修改 BaseDaoImpl 最终生成的 sql 来控制，但需要在ervice 层获取用户信息来做判断

可通过dubbo Filter 提供的隐式传参

1. getaway 中代码Demo　com.lezhi.axy.gateway.common.DubboFilter
2. service中 接收的代码demo(已实现aop)　com.lezhi.axy.service.aop.DataFilterAspect

以上代码仅做了简单的参数传递和AOP拦截，因为需要改动原有的设计，目前没有去控制数据，后续时间可以在此基础上进行改进。

warning注：可能会涉及到有些数据需要权限控制，有些数据不需要权限控制、或者控制权限的段不同的清空。可提供的建议方案：通过自定义注解的方式，仅对添加了自定义注解的方法进行数据的过滤。

## 2.5 关于web安全

● 2.5.1 可能存在的越权问题：

　● 目前的设计可能存在越权操作数据的问题，就算做了数据权限，不同的角色显示不同的数据，是如果编辑了某一条数据，可以在浏览器完整的看到修改接口的接口地址和 参数，那么 如果改了这参数，比如某条记录的id ，就会存在将指定id 的数据进行了编辑操作，而这条数据可能非当前用户限范围。

比如 上面这个是某功能修改数据的接口，如果 有人把这个参数列表中的id ，改为其他的id ，那么就对其他的数据 进行了修改，甚至可能威胁到整个表的数据

　● 可选的解决方案： 敏感操作 前后台要加双重验证

● 2.5.2 可能存在的数据库漏洞

　● 前端校验数据要跟数据库类型、长度进行同步，防止出现前端校验通过但是数据库没有进行校而出现的服务器报错问题 ==比如 某字段A 在数据库 是int 类型，长度是10 ，那么 前端校验的时候要 校验是int 类型，长度不能超过10==

● 2.5.3 弃用代码的删除

　● 由于业务的变化，可能部分功能已经弃用，我们需要对这部分代码进行清除（否则可能由于放维护而没有清除导致的服务器错误，甚至成为系统可能被攻击的漏洞）

● 2.5.4 警惕前端页面的控制

● 在web中，任何只在前端做的控制，都需要警惕，看是否有被修改的可能。下面举例说明

1. 比如权限控制到按钮，有一些做法是从后台返回用户的权限之后，在前端控制是否显示某个钮（如果在页面修改了判断条件，就直接越过限制了）

2. 参考下图

3. 敏感信息切忌直接存储在cookies 中.原因同上，目前我们用的vue ，可以存储在vuex 中coodes 中存储的必要信息，尽量进行加密处理

# 2.6 关于数据稳定安全

● 2.6.1 数据库原子性 操作

● 不做事务处理会导致的问题：数据一致性被破坏——相关联的数据，本应两条都操作成功，但如果不做事务处理，可能 第一条操作完后 出现各种异常，导致 第二条数据没有做同步的操作，最终两条 关联的数据变得不一致。

● 目前我们很多数据库操作没有进行事务管理，可能会导致数据错乱的问题（某些功能已实现，以参考进行，比如com.lezhi.axy.service.impl.api.DormitoryServiceImpl 中通过 @Transactional 解去控制了数据的原子性操作）

● 目前若要处理事务，需要做的改变：

首先，由于我们是微服务架构，由于 微服务的特殊性，对于 一个接口中多次数据库操作，必需要迁到service 中进行，在sercice 中才可以进行事务处理，getaway 仅需要调用 service ，多次操作数据都通过 service 中调不同的dao 层来实现，然后service 方法加上 注解进行事务处理