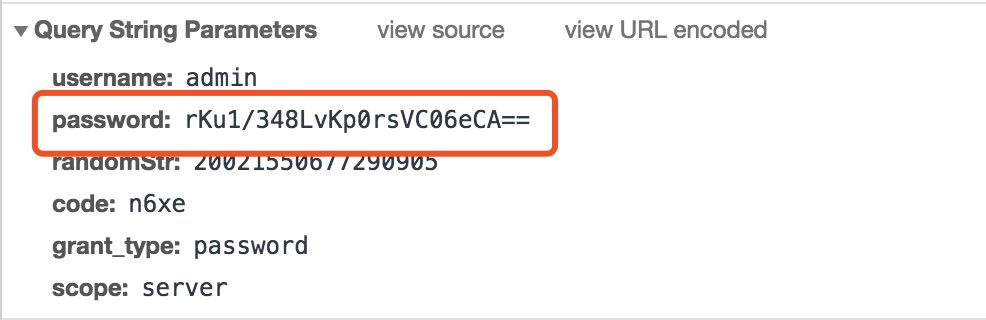
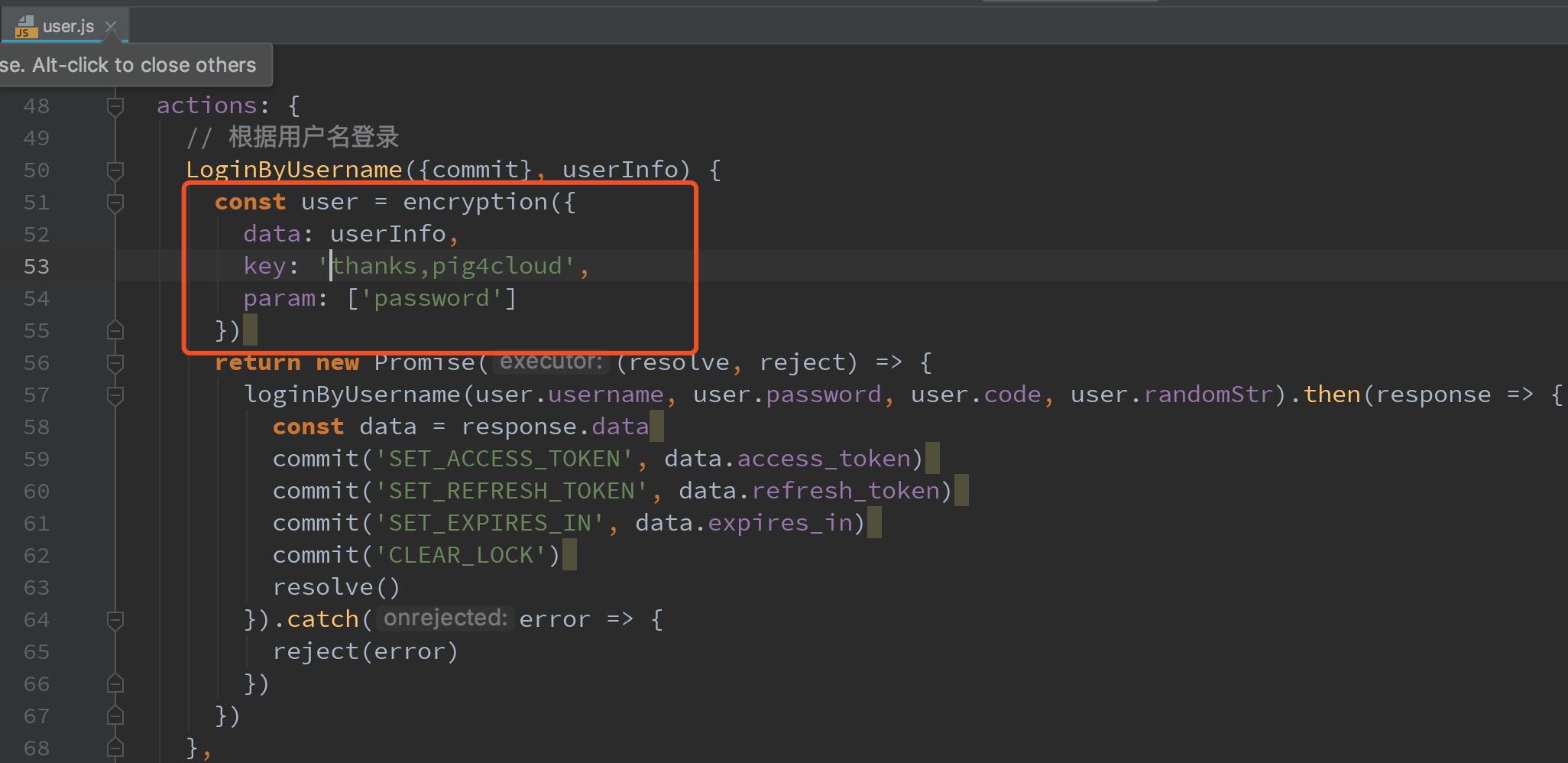
**[#](https://pig4cloud.com/" \l "%E5%89%8D%E7%AB%AF%E6%8A%A5%E6%96%87%E5%8A%A0%E5%AF%86%E7%9A%84%E4%B8%9A%E5%8A%A1) 前端报文加密的业务**

用户登录时，对登录密码进行加密传输



[**#**](https://pig4cloud.com/#%E5%89%8D%E7%AB%AF%E5%8A%A0%E5%AF%86%E5%8A%9F%E8%83%BD) **前端加密功能**

前端提供简单的AES对称加密算法，注意key 和后端网关配置相同，这里打包混淆后，相对安全。 

[**#**](https://pig4cloud.com/#%E5%90%8E%E7%AB%AF%E8%A7%A3%E5%AF%86%E5%8A%9F%E8%83%BD) **后端解密功能**

使用hutool提供的工具类进行解密

public class PasswordDecoderFilter extends AbstractGatewayFilterFactory {

@Override

public GatewayFilter apply(Object config) {

return (exchange, chain) -> {

ServerHttpRequest request = exchange.getRequest();

URI uri = exchange.getRequest().getURI();

String queryParam = uri.getRawQuery();

Map<String, String> paramMap = HttpUtil.decodeParamMap(queryParam, CharsetUtil.UTF\_8);

String password = paramMap.get(PASSWORD);

if (StrUtil.isNotBlank(password)) {

try {

password = decryptAES(password, encodeKey);

} catch (Exception e) {

log.error("密码解密失败:{}", password);

return Mono.error(e);

}

paramMap.put(PASSWORD, password.trim());

}

URI newUri = UriComponentsBuilder.fromUri(uri)

.replaceQuery(HttpUtil.toParams(paramMap))

.build(true)

.toUri();

ServerHttpRequest newRequest = exchange.getRequest().mutate().uri(newUri).build();

return chain.filter(exchange.mutate().request(newRequest).build());

};

}

}

[**#**](https://pig4cloud.com/#%E6%80%BB%E7%BB%93) **总结**

杠精不要抬杠AES 是对称加密，不安全。