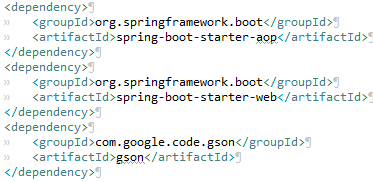
# Spring Boot使用AOP

AOP的强大毋庸置疑，一个切面有5种通知，前置、后置、后置返回、后置异常、环绕。

Talk is cheap.Show U my code！



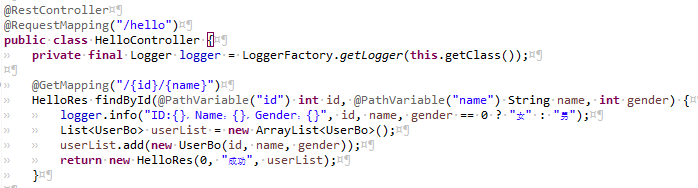
## pom.xml



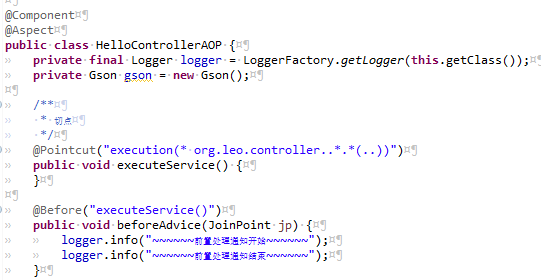
因为个人比较喜欢Gson，所以用这个，不过实际工作中还是推荐阿里的fastjson。

## 前置

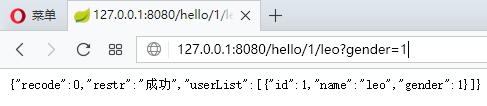
Controller代码



AOP代码



页面



日志

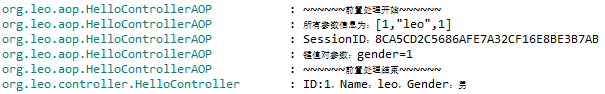


在实际工作中，前置通知可以做许多事情，比如通过Session校验用户是否登录，登录的用户是否有权限，无状态的请求可以校验Token，或者通过全局ID保证接口幂等性等。

修改一下前置通知

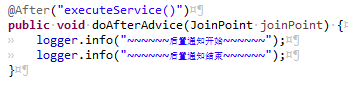


日志

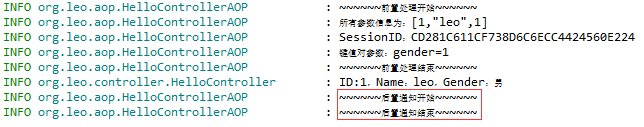


## 后置

AOP代码

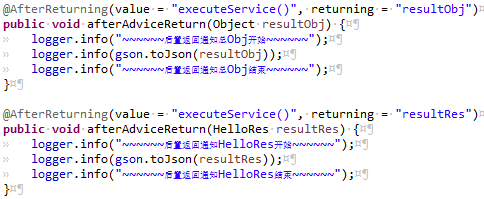


日志

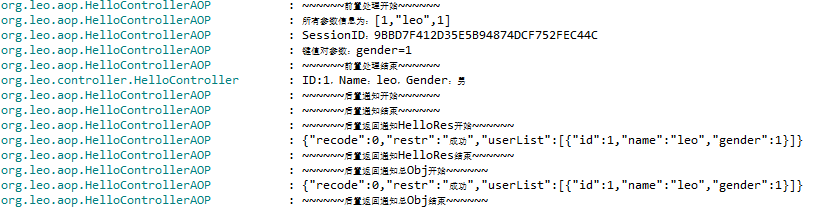


## 后置返回

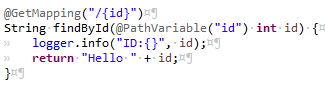
AOP代码



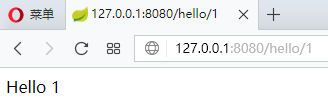
日志



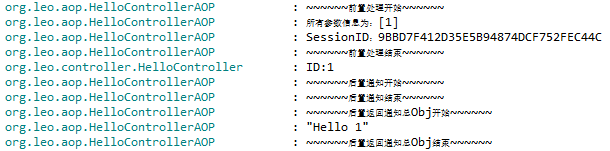
新增一个Controller方法



页面



日志



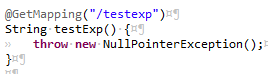
注意对比二者后置返回通知日志的不同。

第一个请求，返回了HelloRes类，所以AOP参数为Object、HelloRes的两个后置返回通知都执行了。

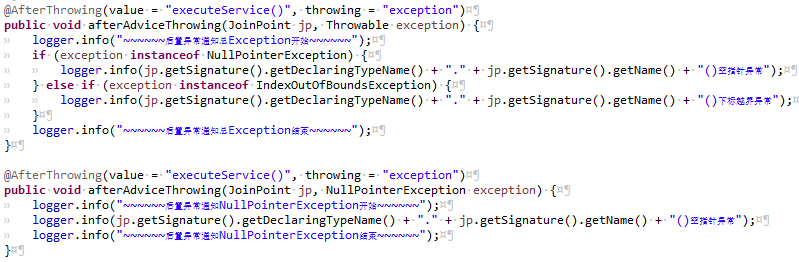
第二个请求，返回String，所以AOP只执行了参数为Object的后置返回通知。

## 后置异常

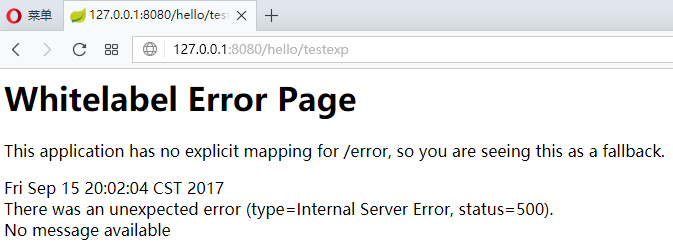
Controller代码



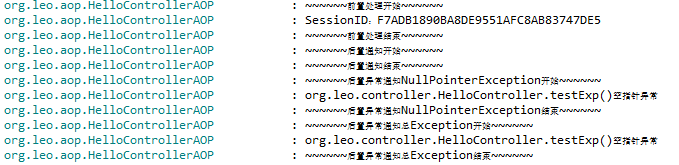
AOP代码



页面

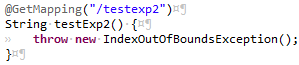


日志

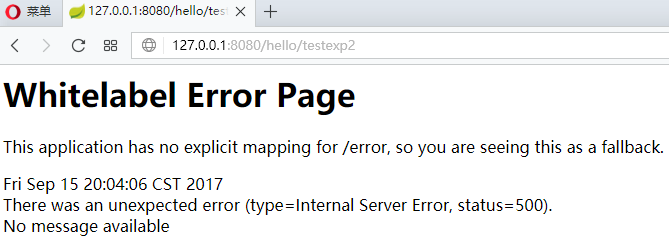


因为发生了异常，所以后置返回通知不执行了。

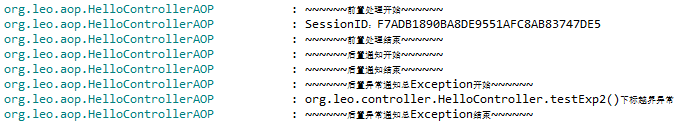
新增一个Controller方法



页面



日志



注意对比二者后置异常通知日志的不同。

第一个请求，返回了空指针异常，所以AOP参数为Throwable、NullPointerException的两个后置异常通知都执行了。

第二个请求，返回下标越界异常，所以AOP只执行了参数为Throwable的后置异常通知。

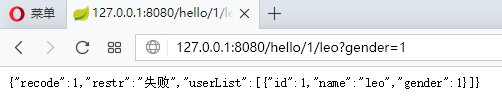
## 环绕

这个才最强大！

AOP代码

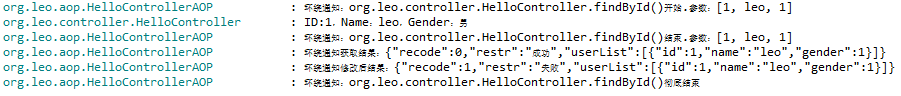


页面

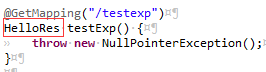


注意返回的recode和restr与之前的不同，里面的值被AOP修改了。

日志



把Controller之前抛异常的方法修改返回值



页面



日志



这其实给了我们一个思路：把所有返回数据用一个通用的类封装起来，通过环绕通知进行统一处理，这样我们的业务开发人员就不必考虑异常处理的事情了。