统一处理异常，与记录日志

@ControllerAdvice

-用于修饰类，表示该类是Controller的全局配置类

-在此类中，可以对Controller进行如下三种全局配置

：异常处理方案，绑定数据方案，绑定参数方案

@ExceptionHandler

用于修饰方法，该方法会在Controller出现异常后被调用，用于处理捕获到的异常。

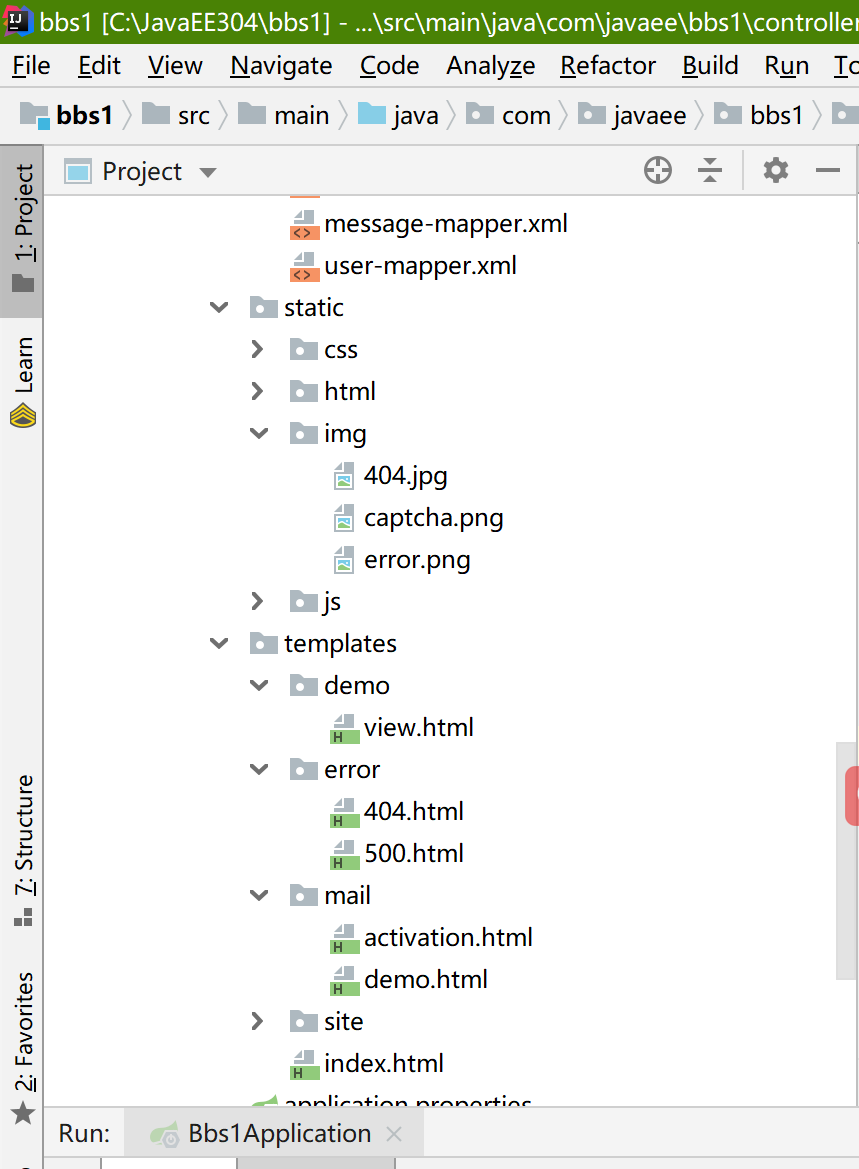
@ModelAttribute

用于修饰方法，该方法会在Controller方法执行前被调用，用于为Model对象绑定参数。

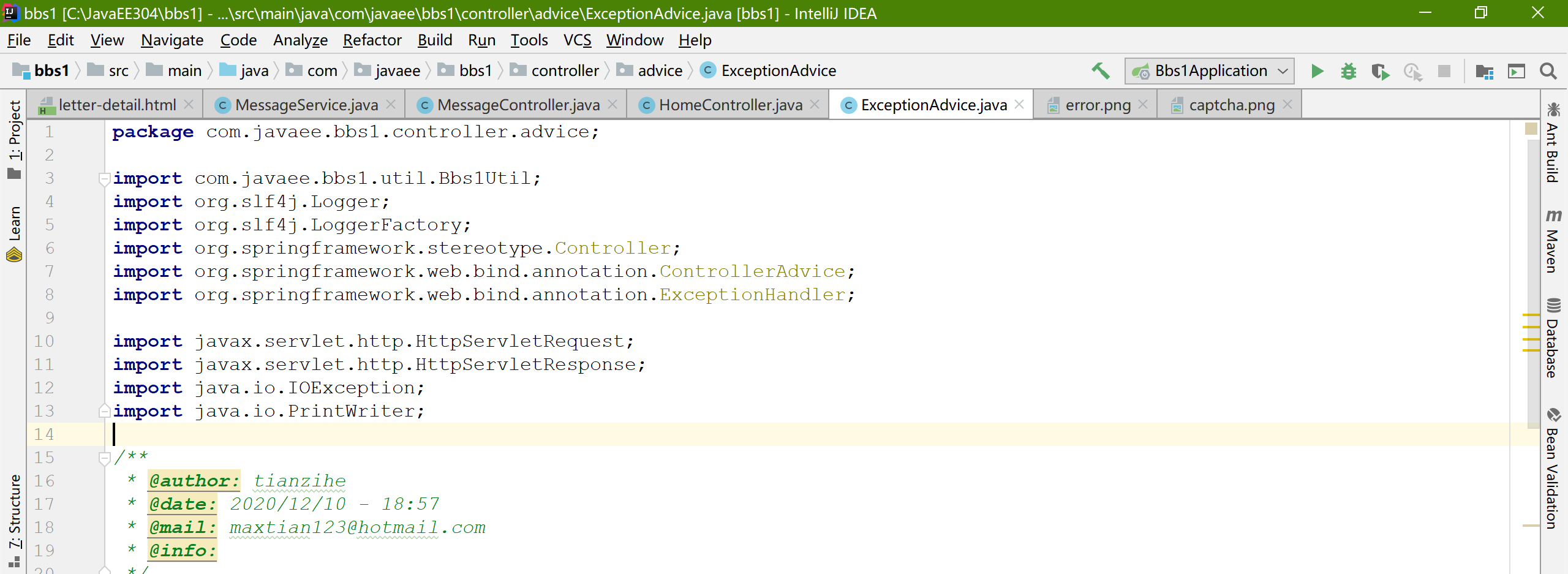
@DataBinder

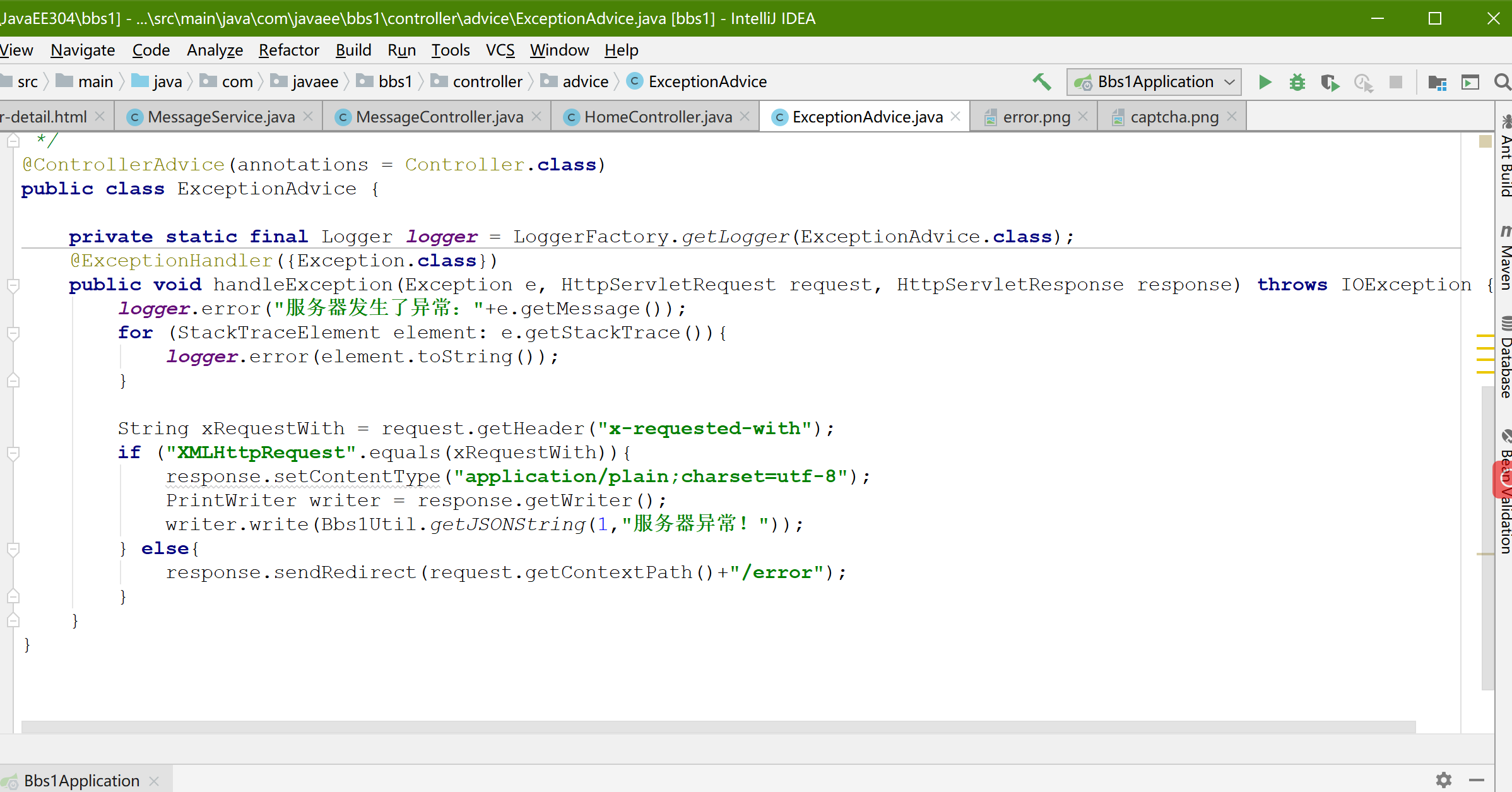
用于修饰方法，该方法会在Controller执行前被调用，用于绑定参数的转换器。

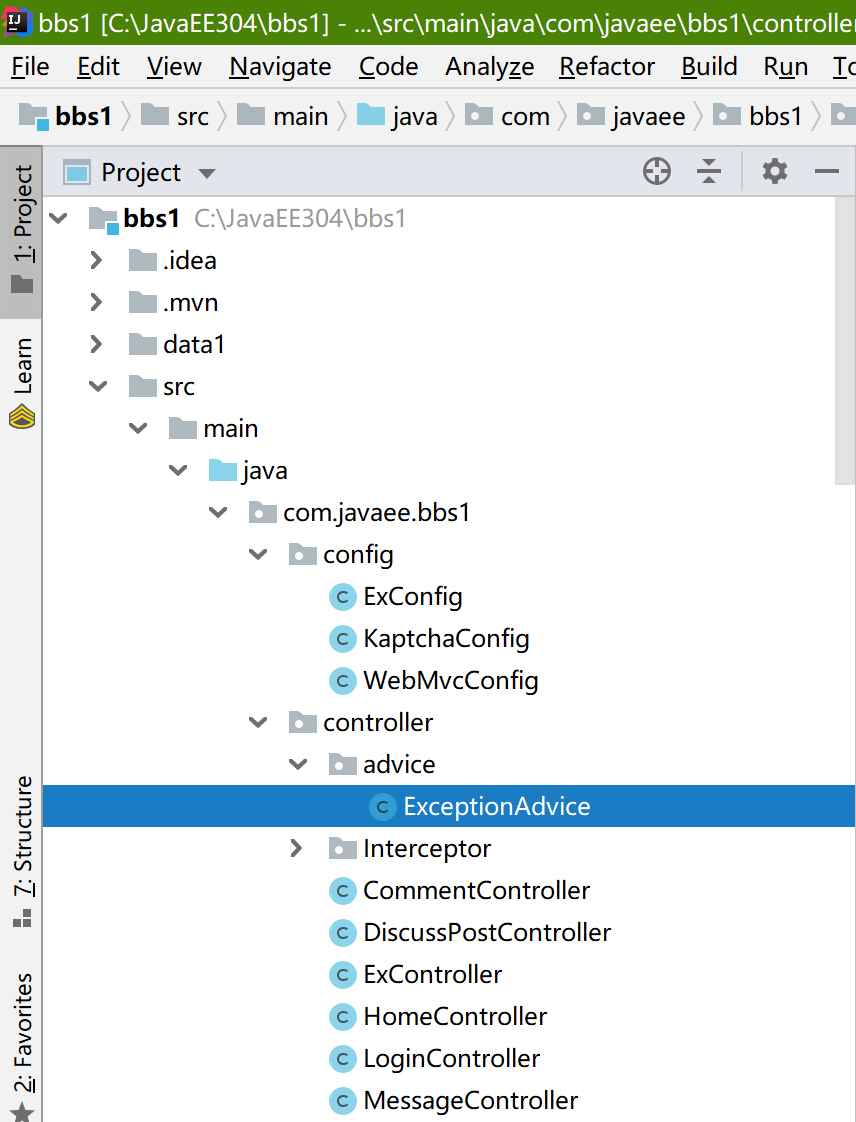
按图里位置加入文件，如出错时图片不显示，则清除浏览器缓存。



Controller包下加入advice包下加入ExceptionAdvice类



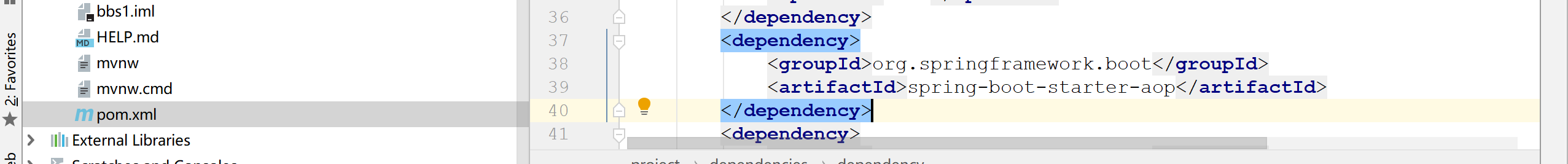




如果异常会出现服务器异常的结果，增加用户体验

统一记录日志

pom文件导入Aop的依赖



<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-aop</**artifactId**>  
</**dependency**>

AOP

Aspect Oriented Programing

即面向切面编程

AOP是一种编程思想，是对OOP(面向对象)的补充。

可以进一步提高编程效率。

对于帖子，评论，消息等模块进行

权限检查

记录日志

事务管理

Weaving

1.编译时织入，要特殊编译器

2.装载。。。。。。。。。。。。。。。类装载器

3.运行时。。。。。。。。。。。。。为目标生成代理对象

目标对象 target

可织入的点Joinpoint.

Pointcut 要将那些代码织入到那些位置

Advice通知实现具体的系统逻辑

实现在代码前，后，返回，异常？

AOP的实现

AspectJ

-AspectJ 是语言级的实现，它扩展了java语言，定义了AOP语法

-编译时织入代码，有一个专门的编译器

Spring AOP

-Spring AOP使用纯Java实现，它不需要专门的编译过程，也不需要特殊的类装载器

-SpringAOP 在运行时通过代理方式织入代码，只支持方法类型的链接点。

-Spring支持对AspectJ的集成

JDK动态代理

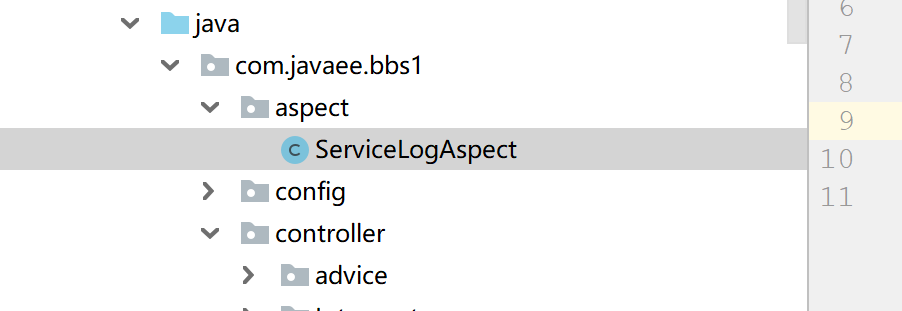
需要接口，在接口织入代码

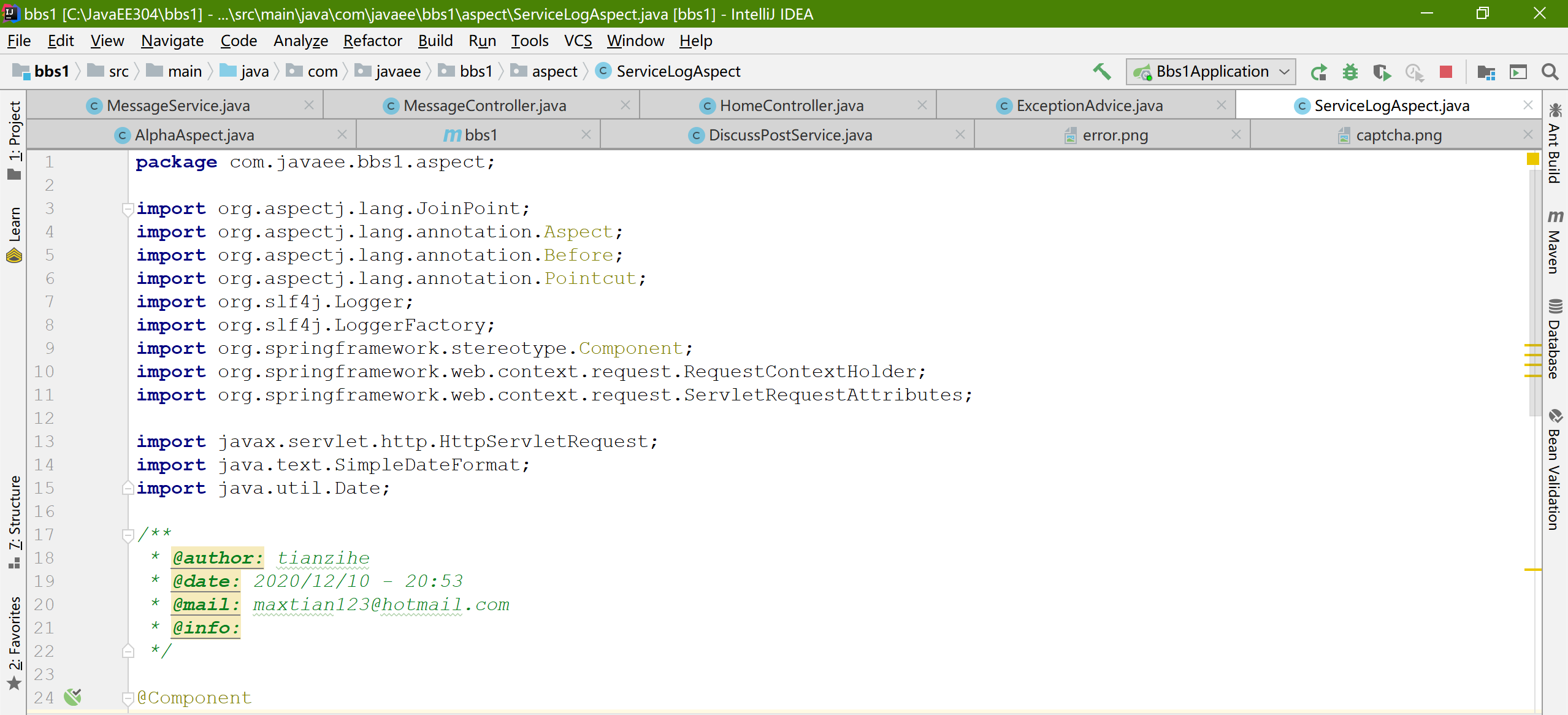
CGLib动态代理

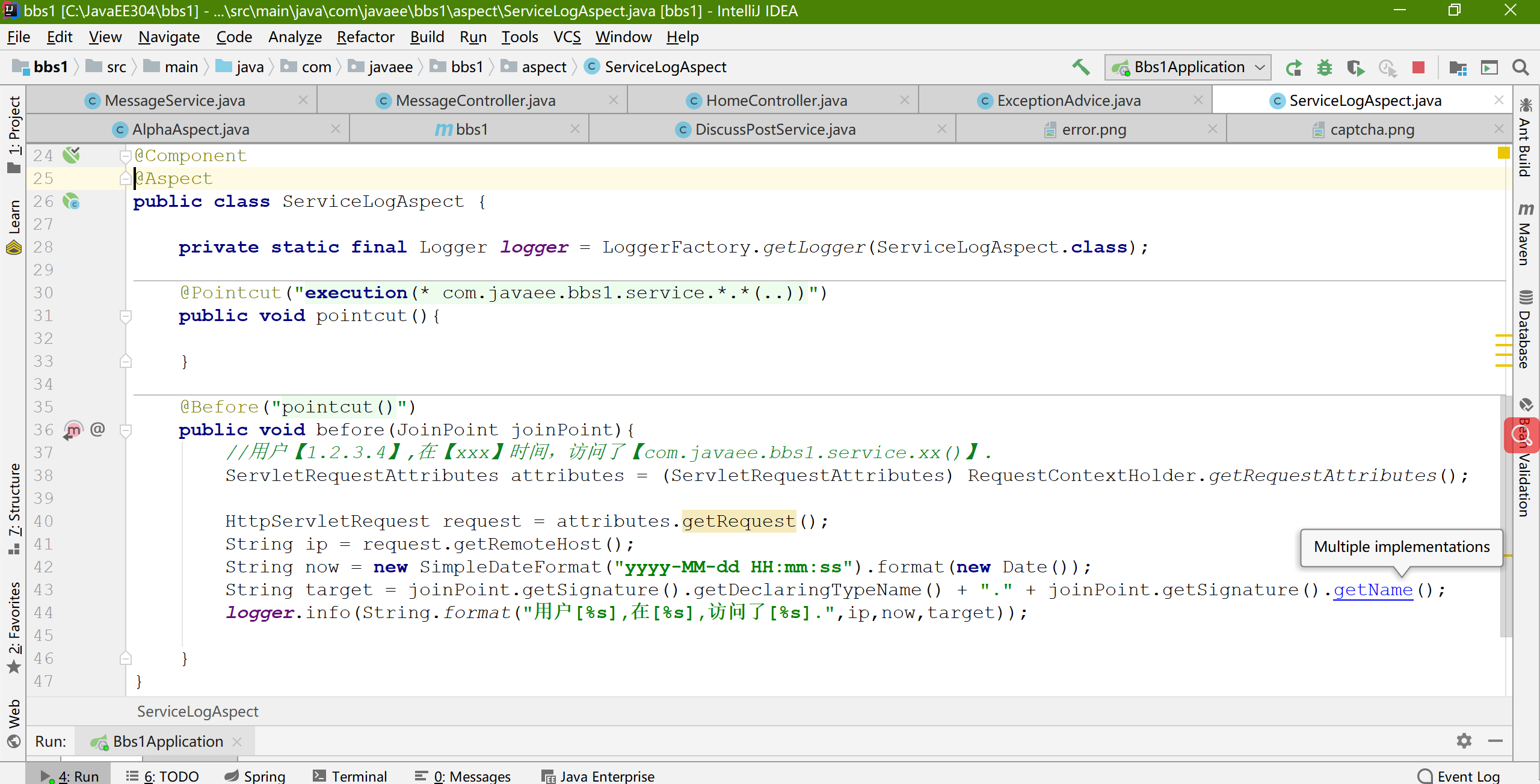
采用底层字节码技术

无接口，在子类实例中织入代码

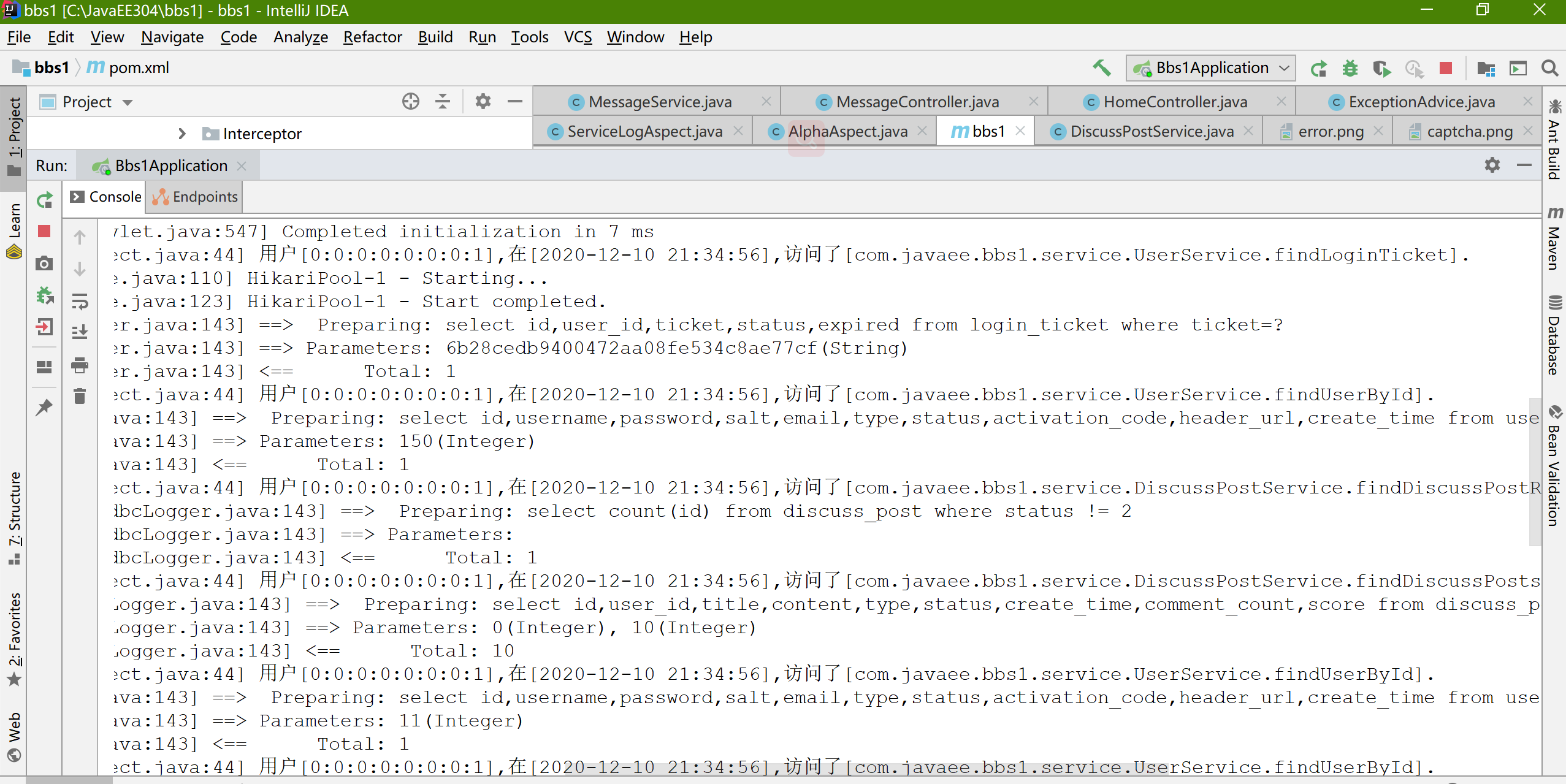
新建类ServiceLogAspect







结果运行访问网页，日志结果为



这样运作流程看起来就更清晰了