36_高阶实战：基于 Spring AOP 机制拦截处理登陆信息

1、开篇

上节课验证了处理异常的代码，通过 GlobalExceptionHandler 类处理 Controller 中抛出的异常信息，从而解决页面报错的问题。本节课通过使用 Spring AOP 的方式拦截登陆信息，并且判断今天的内容：

添加 LoginInterceptor 类用来拦截登陆信息

配置 spring-web.xml 中的拦截器

2、添加 LoginInterceptor 类拦截登陆信息

既然是拦截登陆信息，就需要先建立一个拦截类，用来处理登陆请求之前的判断工作。如图 1 所示，在/spring/interceptor 目录下面建立 LoginInterceptor 类，这个类是专门用来拦截登陆请求的。该类实现了 HandlerInterceptor，并且 Override 了 preHandle 方法，也就是在处理登陆请求之前会执行该方法。下面来看下方法中执行的内容。

图 1 加入 LoginInterceptor 类

图 1 的 preHandle 方法中传入了 request 参数，其类型是 HttpServletRequest 类型，另外还有 response 其类型是 HttpServletResponse，最后是 Object 类型的 handler。

首先，从 request 中也就是请求中通过 getParameter 获取参数，这里的参数是 phoneNumber 也就是手机号。

然后判断手机号是否为空，再通过 getHeader 方法获取 phoneNumber，看看在请求头中是否存在手机号。如果也不存在，那么打印日志说明请求接口不包含手机号。

接下来，通过 response 的 getOutputStream 获取相应的 response 流。通过 setHeader 方法设置返回的类型 content-type 和 application/json。另外通过指定 JSON 字符串中的编码格式 UTF8，告知相应的编码模式。

最后通过，outputStream 的 write 方法返回输出的字节流，如果在登陆的时候不存在手机号信息就会返回失败，否则可以登陆继续后面的操作。

3、配置 spring-web.xml 中的拦截器

看完了登陆拦截器的代码部分，再回到 spring-web.xml 配置文件中，来看看配置部分。这里会在 interceptors 加入一个 interceptors（拦截器）。如图 2 所示，这个拦截器包括以下几个部分：

Mapping：这里主要指定需要拦截的路径，包含在该配置下的路径会被拦截到，否则拦截器是不会处理的。这里设置的是“/**”，意思是全部路径都会拦截。

Exclude-mapping：这里配置除了那些路径是不做拦截的。也是与 Mapping 配置相呼应的一个配置，也就是说拦截器也有例外，在这里配置的路径不做拦截处理。这里配置了三个节点，包括“/consumer/login”、”/demo/**”、”/”。

Bean：这里是对拦截器进行指定，也就是说上面的规则通过哪个拦截器实现。这里我们配置了“com.ruyuan.little.project.spring.interceptor.LoginInterceptor"，也就是我们刚刚编写的拦截器。

”

图 2 spring-web.xml 配置文件

4、总结

本节通过 Spring AOP 实现了登陆信息拦截的功能，该功能包括两个部分：一、编写拦截器类，其中包括拦截规则以及拦截以后的处理；二、为 spring-web.xml 中配置拦截器的信息，拦截路径、例外拦截路径以及拦截器的实现类。鉴于本节课是这周最后一节课，在这里将本周课程做一个总结。

本周以 Spring AOP 在项目中的应用为主，通过 logback 框架为切入点引出 Spring AOP 拦截并打印日志的功能，同时提出处理异常的问题，又使用 Spring AOP 处理系统的全局异常，最后通过拦截器的方式判断登陆时信息是否合乎要求。下周课将专注于另外一个系统级别的功能介绍：数据库访问。这里将代码给大家，下期见，拜拜。

友情提示：本章讲述代码只是部分核心代码，完整代码请查阅文末链接中代码，谢谢。