Springboot集成拦截器

# 注册拦截器

## 代码实现

@Configuration

public class InterceptorConfig implements WebMvcConfigurer {

@Bean

public HandlerInterceptor getTemplateInterceptor(){

return new TemplateInterceptor();

}

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(getTemplateInterceptor()).addPathPatterns("/\*\*");

}

}

## 1.2、注意事项

1、注解：Configuration，在springboot启动时加载

2、addPathPatterns：拦截指定请求路径

# 2、自定义拦截器

## 2.1、代码实现

public class TemplateInterceptor implements HandlerInterceptor {

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

if (!(handler instanceof HandlerMethod)) {

return true;

}

HandlerMethod handlerMethod = (HandlerMethod) handler;

Method method = handlerMethod.getMethod();

if(method.getAnnotation(InterceptorAnnotation.class) != null){

//拦截请求，写业务逻辑

//1、读取Header中指定参数值

request.getHeader("");

//2、读取请求参数中指定参数值

request.getParameter("");

//3、若校验失败，返回前端Json串

response.setStatus(HttpServletResponse.SC\_UNAUTHORIZED);

response.setContentType(MediaType.APPLICATION\_JSON\_VALUE);

PrintWriter out = null;

try {

out = response.getWriter();

out.println(jsonStr);

} catch (Exception e) {

e.printStackTrace();

} finally {

if (null != out) {

out.flush();

out.close();

}

}

}

return true;

}

}

## 2.2、注意事项

1、method.getAnnotation()：拦截方法上是否有指定注解，若有指定注解则进入，否则不拦截

2、request获取请求头和请求参数

3、返回Json字符串

# 3、创建注解

## 3.1、代码实现

//在需要拦截的Controller方法上使用此注解

@Target({ElementType.METHOD})//可用在方法名上

@Retention(RetentionPolicy.RUNTIME)//运行时有效

public @interface InterceptorAnnotation {

}