Spring AOP实现后台管理系统日志管理

7-02-08 15:31:52 张三疯不疯 阅读数 16936 更多

pring AOP实现后台管理系统日志管理

十原则和思路：

元注解方式结合AOP，灵活记录操作日志

能够记录详细错误日志为运维提供支持

日志记录尽可能减少性能影响

义日志记录元注解

```
package com.myron.ims.annotation;

import java.lang.annotation.*;

/**
 * 自定义注解 拦截Controller
 *
 * @author lin.r.x
 *
 */
@Target({ ElementType.PARAMETER, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
public @interface SystemControllerLog {
      /**
    * 描述业务操作 例:Xxx管理-执行Xxx操作
    * @return
    */
      String description() default "";
}
```

义用于记录日志的实体类

```
package com.myron.ims.bean;

import java.io.Serializable;
import com.myron.common.util.StringUtils;
import com.myron.common.util.UuidUtils;
import com.fasterxml.jackson.annotation.JsonFormat;
import java.util.Date;
import java.util.Map;

/**
 * 日志类-记录用户操作行为
 * @author lin.r.x
 *
 */
public class Log implements Serializable{
    private static final long serialVersionUID = 1L;

    private String logId;            //日志主键
    private String type;             //日志类型
    private String title;            //日志标题
    private String remoteAddr;           //请求地址
    private String requestUri;           //URI
    private String method;           //请求方式
    private String params;           //提交参数
    private String exception;            //异常
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss", timezone = "GMT+8")
```

```
    private Date operateDate;            //开始时间
    private String timeout;          //结束时间
    private String userId;           //用户ID

    public String getLogId() {
        return StringUtils.isBlank(logId) ? logId : logId.trim();
    }
    public void setLogId(String logId) {
        this.logId = logId;
    }


    public String getType() {
        return StringUtils.isBlank(type) ? type : type.trim();
    }
    public void setType(String type) {
        this.type = type;
    }


    public String getTitle() {
        return StringUtils.isBlank(title) ? title : title.trim();
    }
    public void setTitle(String title) {
        this.title = title;
    }


    public String getRemoteAddr() {
        return StringUtils.isBlank(remoteAddr) ? remoteAddr : remoteAddr.trim();
    }
    public void setRemoteAddr(String remoteAddr) {
        this.remoteAddr = remoteAddr;
    }


    public String getRequestUri() {
        return StringUtils.isBlank(requestUri) ? requestUri : requestUri.trim();
    }
    public void setRequestUri(String requestUri) {
        this.requestUri = requestUri;
    }


    public String getMethod() {
        return StringUtils.isBlank(method) ? method : method.trim();
    }
    public void setMethod(String method) {
        this.method = method;
    }


    public String getParams() {
        return StringUtils.isBlank(params) ? params : params.trim();
    }
    public void setParams(String params) {
        this.params = params;
    }

    /**
    * 设置请求参数
    * @param paramMap
    */
    public void setMapToParams(Map<String, String[]> paramMap) {
        if (paramMap == null){
            return;
        }
        StringBuilder params = new StringBuilder();
        for (Map.Entry<String, String[]> param : ((Map<String, String[]>)paramMap).entrySet()){
            params.append(("".equals(params.toString()) ? "" : "&") + param.getKey() + "=");
            String paramValue = (param.getValue() != null && param.getValue().length > 0 ? param.getValue()[0] : "");
            params.append(StringUtils.abbr(StringUtils.endsWithIgnoreCase(param.getKey(), "password") ? "" : paramValue, 100));
```

```
        }
        this.params = params.toString();
    }


    public String getException() {
        return StringUtils.isBlank(exception) ? exception : exception.trim();
    }
    public void setException(String exception) {
        this.exception = exception;
    }


    public Date getOperateDate() {
        return operateDate;
    }
    public void setOperateDate(Date operateDate) {
        this.operateDate = operateDate;
    }


    public String getTimeout() {
        return StringUtils.isBlank(timeout) ? timeout : timeout.trim();
    }
    public void setTimeout(String timeout) {
        this.timeout = timeout;
    }


    public String getUserId() {
        return StringUtils.isBlank(userId) ? userId : userId.trim();
    }
    public void setUserId(String userId) {
        this.userId = userId;
    }

}
```

义日志AOP切面类

```
package com.myron.ims.aop;

import java.lang.reflect.Method;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.reflect.MethodSignature;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.NamedThreadLocal;
import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;
import org.springframework.stereotype.Component;

import com.myron.common.util.DateUtils;
import com.myron.common.util.UuidUtils;
import com.myron.ims.annotation.SystemControllerLog;
import com.myron.ims.annotation.SystemServiceLog;
import com.myron.ims.bean.Log;
import com.myron.ims.bean.User;
import com.myron.ims.service.LogService;

```

```java
/**
 * 系统日志切面类
 * @author lin.r.x
 *
 */
@Aspect
@Component
public class SystemLogAspect {
    private  static  final Logger logger = LoggerFactory.getLogger(SystemLogAspect. class);

    private static final ThreadLocal<Date> beginTimeThreadLocal =
            new NamedThreadLocal<Date>("ThreadLocal beginTime");
    private static final ThreadLocal<Log> logThreadLocal =
            new NamedThreadLocal<Log>("ThreadLocal log");

    private static final ThreadLocal<User> currentUser=new NamedThreadLocal<>("ThreadLocal user");

    @Autowired(required=false)
    private HttpServletRequest request;

    @Autowired
    private ThreadPoolTaskExecutor threadPoolTaskExecutor;

    @Autowired
    private LogService logService;

    /**
   * Controller层切点 注解拦截
   */
    @Pointcut("@annotation(com.myron.ims.annotation.SystemControllerLog)")
    public void controllerAspect(){}

    /**
   * 前置通知 用于拦截Controller层记录用户的操作的开始时间
   * @param joinPoint 切点
   * @throws InterruptedException
   */
    @Before("controllerAspect()")
    public void doBefore(JoinPoint joinPoint) throws InterruptedException{
        Date beginTime=new Date();
        beginTimeThreadLocal.set(beginTime);//线程绑定变量（该数据只有当前请求的线程可见）
        if (logger.isDebugEnabled()){//这里日志级别为debug
            logger.debug("开始计时: {}  URI: {}", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS")
                .format(beginTime), request.getRequestURI());
        }

        //读取session中的用户
        HttpSession session = request.getSession();
        User user = (User) session.getAttribute("ims_user");
        currentUser.set(user);

    }

    /**
   * 后置通知 用于拦截Controller层记录用户的操作
   * @param joinPoint 切点
   */
    @SuppressWarnings("unchecked")
    @After("controllerAspect()")
    public void doAfter(JoinPoint joinPoint) {
        User user = currentUser.get();
        if(user !=null){
            String title="";
            String type="info";                         //日志类型(info:入库,error:错误)
            String remoteAddr=request.getRemoteAddr();//请求的IP
            String requestUri=request.getRequestURI();//请求的Uri
            String method=request.getMethod();          //请求的方法类型(post/get)
            Map<String,String[]> params=request.getParameterMap(); //请求提交的参数

            try {
                title=getControllerMethodDescription2(joinPoint);
            } catch (Exception e) {
```

```
            e.printStackTrace();
        }
        // 打印JVM信息。
        long beginTime = beginTimeThreadLocal.get().getTime();//得到线程绑定的局部变量（开始时间）
        long endTime = System.currentTimeMillis();  //2、结束时间
        if (logger.isDebugEnabled()){
            logger.debug("计时结束：{} URI: {} 耗时： {} 最大内存: {}m 已分配内存: {}m 已分配内存中的剩余空间: {}m 最大可用内存: {}m",
                    new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS").format(endTime),
                    request.getRequestURI(),
                    DateUtils.formatDateTime(endTime - beginTime),
                    Runtime.getRuntime().maxMemory()/1024/1024,
                    Runtime.getRuntime().totalMemory()/1024/1024,
                    Runtime.getRuntime().freeMemory()/1024/1024,
                    (Runtime.getRuntime().maxMemory()-Runtime.getRuntime().totalMemory()+Runtime.getRuntime().freeMemory(
        }

        Log log=new Log();
        log.setLogId(UuidUtils.creatUUID());
        log.setTitle(title);
        log.setType(type);
        log.setRemoteAddr(remoteAddr);
        log.setRequestUri(requestUri);
        log.setMethod(method);
        log.setMapToParams(params);
        log.setUserId(user.getId());
        Date operateDate=beginTimeThreadLocal.get();
        log.setOperateDate(operateDate);
        log.setTimeout(DateUtils.formatDateTime(endTime - beginTime));

        //1.直接执行保存操作
        //this.logService.createSystemLog(log);

        //2.优化:异步保存日志
        //new SaveLogThread(log, logService).start();

        //3.再优化:通过线程池来执行日志保存
        threadPoolTaskExecutor.execute(new SaveLogThread(log, logService));
        logThreadLocal.set(log);
    }

}

/**
* 异常通知 记录操作报错日志
* @param joinPoint
* @param e
*/
@AfterThrowing(pointcut = "controllerAspect()", throwing = "e")
public  void doAfterThrowing(JoinPoint joinPoint, Throwable e) {
    Log log = logThreadLocal.get();
    log.setType("error");
    log.setException(e.toString());
    new UpdateLogThread(log, logService).start();
}

/**
* 获取注解中对方法的描述信息 用于service层注解
* @param joinPoint切点
* @return discription
*/
public static String getServiceMthodDescription2(JoinPoint joinPoint) {
    MethodSignature signature = (MethodSignature) joinPoint.getSignature();
    Method method = signature.getMethod();
    SystemServiceLog serviceLog = method
            .getAnnotation(SystemServiceLog.class);
    String discription = serviceLog.description();
    return discription;
}

/**
* 获取注解中对方法的描述信息 用于Controller层注解
*
* @param joinPoint 切点
```

```
     * @return discription
     */
    public static String getControllerMethodDescription2(JoinPoint joinPoint) {
        MethodSignature signature = (MethodSignature) joinPoint.getSignature();
        Method method = signature.getMethod();
        SystemControllerLog controllerLog = method
                .getAnnotation(SystemControllerLog.class);
        String discription = controllerLog.description();
        return discription;
    }

    /**
     * 保存日志线程
     */
    private static class SaveLogThread implements Runnable {
        private Log log;
        private LogService logService;

        public SaveLogThread(Log log, LogService logService) {
            this.log = log;
            this.logService = logService;
        }

        @Override
        public void run() {
            logService.createLog(log);
        }
    }

    /**
     * 日志更新线程
     */
    private static class UpdateLogThread extends Thread {
        private Log log;
        private LogService logService;

        public UpdateLogThread(Log log, LogService logService) {
            super(UpdateLogThread.class.getSimpleName());
            this.log = log;
            this.logService = logService;
        }

        @Override
        public void run() {
            this.logService.updateLog(log);
        }
    }
}
```

pring 配置扫描切面，开启@AspectJ注解的支持

```
<!-- 启动对@AspectJ注解的支持 -->
<aop:aspectj-autoproxy/>
<!-- 扫描切点类组件 -->
<context:component-scan base-package="com.myron.ims.aop" />
<context:component-scan base-package="com.myron.ims.service"/>

<bean id="taskExecutor"  class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
    <property name="corePoolSize" value="5" />
    <property name="maxPoolSize" value="10" />
    <property name="WaitForTasksToCompleteOnShutdown" value="true" />
</bean>
```

用范例LoginController方法中添加日志注解

```
    /**
     *系统登入
     */
    @RequestMapping("/login.do")
    @SystemControllerLog(description="登入系统")
    @ResponseBody
```

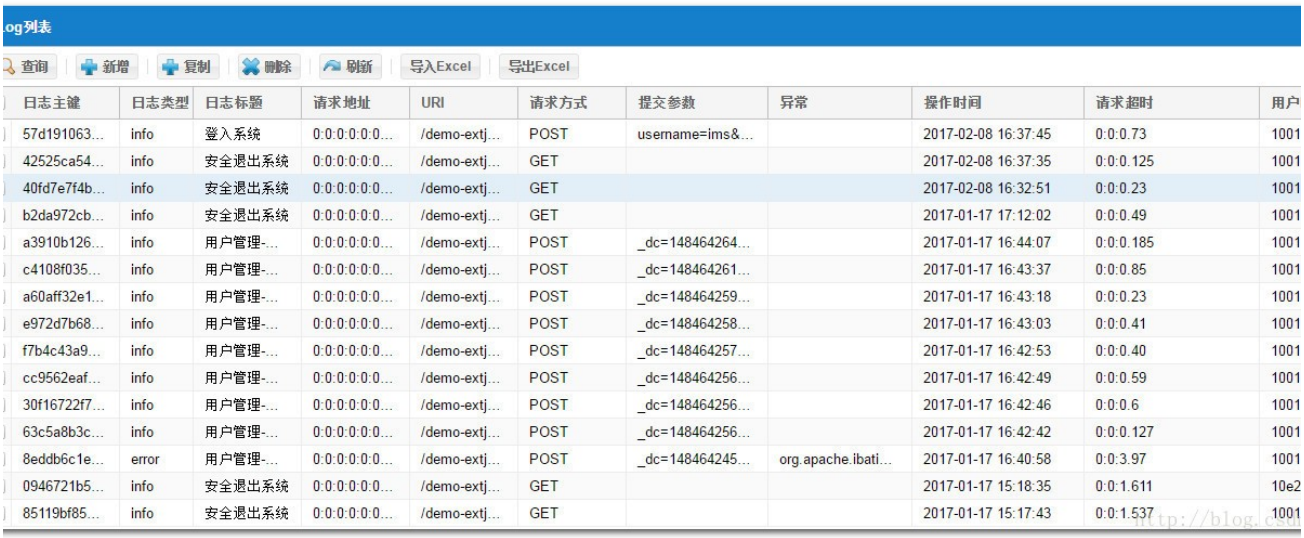
```
    public Map<String, Object> login(String username, String password, Boolean rememberMe, HttpServletRequest req){
    //业务代码省略...
}

    /**
   * 安全退出登入
   * @return
   */
    @SystemControllerLog(description="安全退出系统")
    @RequestMapping("logout.do")
    public String logout(){
        Subject subject=SecurityUtils.getSubject();
        if(subject.isAuthenticated()){
            subject.logout(); // session 会销毁，在SessionListener监听session销毁，清理权限缓存
        }
        return "/login.jsp";
    }
```

9

32

运行效果



log列表

查询 | 新增 | 复制 | 删除 | 刷新 | 导入Excel | 导出Excel

日志主键	日志类型	日志标题	请求地址	URI	请求方式	提交参数	异常	操作时间	请求超时	用户
57d191063...	info	登入系统	0:0:0:0:0:0...	/demo-extj...	POST	username=ims&...		2017-02-08 16:37:45	0:0:0.73	1001
42525ca54...	info	安全退出系统	0:0:0:0:0:0...	/demo-extj...	GET			2017-02-08 16:37:35	0:0:0.125	1001
40fd7e7f4b...	info	安全退出系统	0:0:0:0:0:0...	/demo-extj...	GET			2017-02-08 16:32:51	0:0:0.23	1001
b2da972cb...	info	安全退出系统	0:0:0:0:0:0...	/demo-extj...	GET			2017-01-17 17:12:02	0:0:0.49	1001
a3910b126...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464264...		2017-01-17 16:44:07	0:0:0.185	1001
c4108f035...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464261...		2017-01-17 16:43:37	0:0:0.85	1001
a60aff32e1...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464259...		2017-01-17 16:43:18	0:0:0.23	1001
e972d7b68...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464258...		2017-01-17 16:43:03	0:0:0.41	1001
f7b4c43a9...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464257...		2017-01-17 16:42:53	0:0:0.40	1001
cc9562eaf...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464256...		2017-01-17 16:42:49	0:0:0.59	1001
30f16722f7...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464256...		2017-01-17 16:42:46	0:0:0.6	1001
63c5a8b3c...	info	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464256...		2017-01-17 16:42:42	0:0:0.127	1001
8eddb6c1e...	error	用户管理-...	0:0:0:0:0:0...	/demo-extj...	POST	_dc=148464245...	org.apache.ibati...	2017-01-17 16:40:58	0:0:3.97	1001
0946721b5...	info	安全退出系统	0:0:0:0:0:0...	/demo-extj...	GET			2017-01-17 15:18:35	0:0:1.611	10e2
85119bf85...	info	安全退出系统	0:0:0:0:0:0...	/demo-extj...	GET			2017-01-17 15:17:43	0:0:1.537	1001

补充源码地址：https://github.com/MusicXi/demo-aop-log

文章最后发布

有 0 个人打赏

ringMVC 记录用户的操作日志

阅读数 1万+

pringmvc做用户的操作日志记录基本有两种方式1.利用aop做（推荐）2.利用拦截器做（不推荐）AOP做操作日... 博文 来自：sqyNick

梦忆故城 3个月前 #17楼

请问访问网页的用户名和密码是多少啊？

h3455125 4个月前 #16楼

非常棒的框架，很好用。想请教大佬，现在你的框架粒度是方法级的，我现在的需求是要到行级别的，有什么好的建议吗？比如一个方法内部有ABC三块代码，我要在之间插入通知。

卡尔一点也不卡 6个月前 #15楼

代码没有spring配置文件

查看 32 条热评