Spring 中 @Autowired 注解和静态方法关联应用

作者：pplsunny
原文链接：https://ld246.com/article/1544665797803
来源网站：链滴

Spring中@Autowired注解和静态方法关联应用

一、业务场景

spring框架应用中有些静态方法需要依赖被容器管理的类，就像这样：

```
@Component
public class Test {

    @Autowired
    private static UserService userService;

    public static void test() {
        userService.test();
    }
}
```

这样一定会报java.lang.NullPointerException: null异常。

二、原理剖析

静态变量、类变量不是对象的属性，而是一个类的属性，所以静态方法是属于类（class）的，普通方才是属于实体对象（也就是New出来的对象）的，spring注入是在容器中实例化对象，所以不能使用态方法。

而使用静态变量、类变量扩大了静态方法的使用范围。静态方法在spring是不推荐使用的，依赖注入主要目的,是让容器去产生一个对象的实例,然后在整个生命周期中使用他们，同时也让testing工作更容易。

一旦你使用静态方法,就不再需要去产生这个类的实例,这会让testing变得更加困难，同时你也不能为个给定的类，依靠注入方式去产生多个具有不同的依赖环境的实例，这种static field是隐含共享的，且是一种global全局状态，spring同样不推荐这样去做。

在Springframework里，我们是不能@Autowired一个静态变量，使之成为一个Spring bean的。为么？其实很简单，因为当类加载器加载静态变量时，Spring上下文尚未加载。所以类加载器不会在be n中正确注入静态类，并且会失败。

三、解决方法

1、将@Autowire加到构造方法上

```
@Component
public class Test {

    private static UserService userService;

    @Autowired
    public Test(UserService userService) {
        Test.userService = userService;
    }
```

```
    public static void test() {
        userService.test();
    }
}
```

2、用@PostConstruct注解

```
@Component
public class Test {

    private static UserService userService;

    @Autowired
    private UserService userService2;

    @PostConstruct
    public void beforeInit() {
        userService = userService2;
    }

    public static void test() {
        userService.test();
    }
}
```

四、真实案例

代码如下：

```
package com.lxxfjr.microbank.integration.gateway;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.annotation.PostConstruct;
import javax.servlet.http.HttpServletRequest;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import com.lxxfjr.microbank.common.util.StringUtil;
import com.lxxfjr.microbank.common.util.security.GateWayRSACoder;
import com.lxxfjr.microbank.dal.daointerface.ApplicationIPWhiteListDAO;
import com.lxxfjr.microbank.dal.daointerface.ExternalKeyInfoDAO;
import com.lxxfjr.microbank.dal.daointerface.SysParamDAO;
import com.lxxfjr.microbank.dal.dataobject.ApplicationIPWhiteListDO;
import com.lxxfjr.microbank.dal.dataobject.ExternalKeyInfoDO;
import com.lxxfjr.microbank.dal.dataobject.SysParamDO;
import com.yjf.common.log.Logger;
import com.yjf.common.log.LoggerFactory;

/**
 * 网关相关验证类
```

```
 *
 * @author lujl
 * @date 2018-12-12
 */
@Component
public class GateWayValidationUtil {

    private static GateWayValidationUtil gateWay;

    @Autowired
    protected SysParamDAO sysParamDAO;

    @Autowired
    protected ApplicationIPWhiteListDAO applicationIPWhiteListDAO;

    @Autowired
    protected ExternalKeyInfoDAO externalKeyInfoDAO;


    protected final static Logger logger = LoggerFactory.getLogger(GateWayValidationUtil.clas
);

    @PostConstruct
    public void init() {
        gateWay = this;
        gateWay.sysParamDAO = this.sysParamDAO;
        gateWay.applicationIPWhiteListDAO = this.applicationIPWhiteListDAO;
        gateWay.externalKeyInfoDAO = this.externalKeyInfoDAO;
    }

    /**
     * IP地址白名单验证
     *
     * @param partnerNo
     *          合作方编号
     * @param ipAddress
     *          IP地址
     * @return {@code true} 允许;{@code false} 不允许;
     */
    public static boolean isIPPermitConnection(String partnerNo,String ipAddress) {

        /** 业务逻辑开始 **/
        // 白名单验证：判断是否需要检验，从系统配置表sys_param查询
        SysParamDO sysParamDOTemp = gateWay.sysParamDAO.findById("CHECK_WHITELIST_
ATA");

        logger.info("平台服务器地址:" + ipAddress);

        ApplicationIPWhiteListDO applicationIPWhiteListInfo = new ApplicationIPWhiteListDO();

        // 开启白名单验证的场合-平台参数【YES】
        if (StringUtil.isNotEmpty(checkWhiteListDate) && checkWhiteListDate.equals("YES")) {

            // 只找同平台下的
```

```
            applicationIPWhiteListInfo.setSource(partnerNo);

            //查询external_server_ip_white_info外部白名单表，首先得有至少一条配置
            long count = gateWay.applicationIPWhiteListDAO
                    .findApplicationIPWhiteListCountByCondition(applicationIPWhiteListInfo, null, nul
);
            if(count==0) {
                return false;
            }

            List<ApplicationIPWhiteListDO> applicationIPWhiteListDOs = gateWay.applicationIP
hiteListDAO
                    .findApplicationIPWhiteListInfoByCondition(applicationIPWhiteListInfo, 0, count,
ull, null);
        }else {
            //未开启验证的场合，直接允许通过
            return true;
        }

    }

    /**
     * 业务数据一致性验证
     * @param request
     *            请求体
     * @return  {@code true} 一致;{@code false} 不一致;
     */
    public static boolean dataConsistencyCheck(HttpServletRequest request) {

        Map<String, String> map = new HashMap<String, String>();

        //业务数据不存在直接返回错误
        if(bizdata==null || "".equals(bizdata)){
            return false ;
        }

        // 检索外部接口验证键表，获取给平台分配的公钥密钥
        ExternalKeyInfoDO externalKeyInfoDO = null;
        try {
            externalKeyInfoDO = gateWay.externalKeyInfoDAO.findByPartnerNo(partnerNo);
        } catch (Exception e) {
            return false;
        }


        // 一致的验签不通过
        if (!isSignOk) {
            logger.error("一致的验签不通过: partnerNo=" + partnerNo);
            return false;
        }else {
            return true;
        }
    }
```

```
}
```

原文链接：Spring 中 @Autowired 注解和静态方法关联应用