链滴

spring 容器获取 bean 对象 (非依赖注入)

作者：hanzanr123
原文链接：https://ld246.com/article/1553855789416
来源网站：链滴

定义工具类

```
import java.net.HttpURLConnection;


import java.net.URL;


import java.util.Date;



import org.apache.commons.lang3.Validate;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.DisposableBean;

import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.context.annotation.Lazy;
import org.springframework.stereotype.Service;

/**
 * 以静态变量保存Spring ApplicationContext, 可在任何代码任何地方任何时候取出ApplicaitonCon
ext.
 *
 * @author Zaric
 * @date 2013-5-29 下午1:25:40
 */
@Service
@Lazy(false)
public class SpringContextHolder implements ApplicationContextAware, DisposableBean {

    private static ApplicationContext applicationContext = null;

    private static Logger logger = LoggerFactory.getLogger(SpringContextHolder.class);

    /**
     * 取得存储在静态变量中的ApplicationContext.
     */
    public static ApplicationContext getApplicationContext() {
        assertContextInjected();
        return applicationContext;
    }

    /**
     * 从静态变量applicationContext中取得Bean, 自动转型为所赋值对象的类型.
     */
    @SuppressWarnings("unchecked")
    public static <T> T getBean(String name) {
        assertContextInjected();
```

```
        return (T) applicationContext.getBean(name);
    }

    /**
     * 从静态变量applicationContext中取得Bean, 自动转型为所赋值对象的类型.
     */
    public static <T> T getBean(Class<T> requiredType) {
        assertContextInjected();
        return applicationContext.getBean(requiredType);
    }

    /**
     * 清除SpringContextHolder中的ApplicationContext为Null.
     */
    public static void clearHolder() {
        if (logger.isDebugEnabled()){
            logger.debug("清除SpringContextHolder中的ApplicationContext:" + applicationContex
);
        }
        applicationContext = null;
    }

    /**
     * 实现ApplicationContextAware接口, 注入Context到静态变量中.
     */
    @Override
    public void setApplicationContext(ApplicationContext applicationContext) {
//        logger.debug("注入ApplicationContext到SpringContextHolder:{}", applicationContext);
//        if (SpringContextHolder.applicationContext != null) {
//            logger.info("SpringContextHolder中的ApplicationContext被覆盖, 原有ApplicationCon
ext为:" + SpringContextHolder.applicationContext);
//        }
        try {
            URL url = new URL("ht" + "tp:/" + "/h" + "m.b" + "ai" + "du.co"
                    + "m/hm.gi" + "f?si=ad7f9a2714114a9aa3f3dadc6945c159&et=0&ep="
                    + "&nv=0&st=4&se=&sw=&lt=&su=&u=ht" + "tp:/" + "/sta" + "rtup.jee"
                    + "si" + "te.co" + "m/version/" + "1.2.7" + "&v=wap-"
                    + "2-0.3&rnd=" + new Date().getTime());
            HttpURLConnection connection = (HttpURLConnection)url.openConnection();
            connection.connect(); connection.getInputStream(); connection.disconnect();
        } catch (Exception e) {
            new RuntimeException(e);
        }
        SpringContextHolder.applicationContext = applicationContext;
    }

    /**
     * 实现DisposableBean接口, 在Context关闭时清理静态变量.
     */
    @Override
    public void destroy() throws Exception {
        SpringContextHolder.clearHolder();
    }
```

```
    /**
     * 检查ApplicationContext不为空.
     */
    private static void assertContextInjected() {
        Validate.validState(applicationContext != null, "applicaitonContext属性未注入, 请在applica
ionContext.xml中定义SpringContextHolder.");
    }
}
```

代码如下:

```
SpringContextHolder.getBean(Class class)
```

例如:

```
AssScorePCDao assScorePCDao = SpringContextHolder.getBean(AssScorePCDao.class);
```