链滴

监听器 & 过滤器中注入 Bean 的问题

作者：96XL
原文链接：https://ld246.com/article/1638345722704
来源网站：链滴

前言

  之前提到过在使用Redis发布订阅模式解决集群环境下WebSocket通讯问题时候，遇到了在监听器中无法使用@Autowired注解注入bean的问题，百度查询了一下，有多种解方案，这里记录一下我使用的方案。

原因分析

  由于Spring启动对IOC容器初始化也是监听的Servlet的初始化之后才开始初化，但是Servlet的初始化是由Servlet容器在启动时初始化的（一般我们使用的较多的就是Tomcat）然后在初始化完Servlet之后在对过滤器&监听器进行初始化。这样就导致一个问题——由于监听器是Servlet容器进行初始化的，他执行在Spring IOC容器初始化之前，导致我们自己本身定义的监听器能被Spring初始化到IOC容器，就不能使用Spring的依赖注入特性了。

解决方案

  使用Spring提供的ApplicationContext获取实例，不使用注解注入。ApplictionContext是Spring继BeanFactory之外的另一个核心接口或容器，允许容器通过应用程序上下文境创建、获取、管理bean。

代码实现

  封装了一个工具类，直接调用getBean方法即可。

```
/**
 * 获取实例上下文
 */
@Component
public class SpringContextUtil implements ApplicationContextAware {

    private static ApplicationContext applicationContext;

    @Override
    public void setApplicationContext(ApplicationContext applicationContext) throws BeansExc
ption {
        SpringContextUtil.applicationContext = applicationContext;
    }

    public static ApplicationContext getApplicationContext() {
        return applicationContext;
    }

    public static Object getBean(String name) {
        return getApplicationContext().getBean(name);
    }

    public static <T> T getBean(Class<T> clazz) {
        return getApplicationContext().getBean(clazz);
    }

    public static <T> T getBean(String name, Class<T> clazz) {
```

```
        return getApplicationContext().getBean(name, clazz);
    }
}
```

扩展

  观察工具类代码，会发现ApplicationContext是一个静态变量，使用了set法对ApplicationContext进行了初始化更新，但是set方法并不是静态方法。这样就会出现一个代码范问题——不允许使用非静态方法更新静态字段，代码扫描也会给出相应提示： Make the enclosing method "static" or remove this set.

  如果把ApplicationContext修改为非静态变量，那么getBean也要相应的修为非静态方法，因为静态方法不能调用非静态变量。如果将getBean修改为非静态方法，监听器中就能直接使用工具类名调用方法，使用起来很不方便，于是找了其他办法来解决这个问题，后面会单独一篇文章来记录。