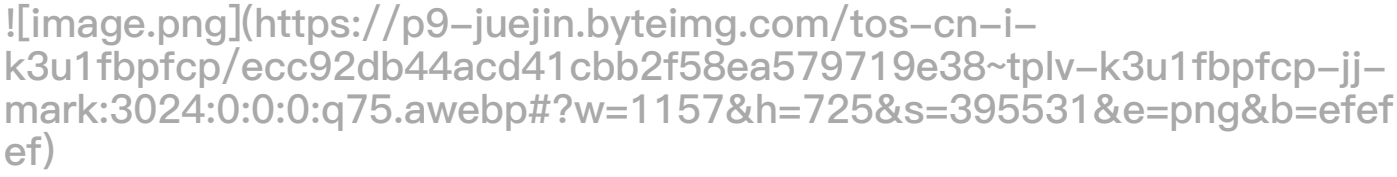
为什么spring 不推荐 @Autowired 用于字段注入?

你好，我是猿java。

作为 Java程序员，Spring绝对是使用频次最高的一个框架，灵活便捷的注入，给开发人员省去了不少的烦恼，今天主要分享一个 Spring 官方不太推荐的方式：@Autowired用于字段注入。

@Autowired警告

从 Spring4.0 开始，官方就不推荐 @Autowire用于字段的注入，如下图，当字段上增加 @Autowired注解时，ideal会出现告警”Field injection is not recommended”，全部内容如下截图：



Spring注入方式

Spring注入有 3种方式：

1. Constructor-based dependency injection 基于构造器注入
2. Setter-based dependency injection 基于set方法注入
3. Field-based dependency injection 基于字段注入

基于构造器注入

基于构造器注入，顾名思义，就是在类的构造器上加@Autowired注解。这种注入方式的主要优点是可以将注入的字段声明为 final，
final修饰的字段是在运行时被初始化，可以直接赋值，也可以在实例构造器中

赋值，赋值后不可修改。下文给出了一个基于构造器注入的例子：

```
@Component
public class ConstructorBasedInjection {

    private final Object object;

    // @Autowired注解可以省去
    @Autowired
    public ConstructorBasedInjection(Object object) {
        this.object = object;
    }
}
```

基于setter方法注入

基于set方法注入，即在 setter方法上加 @Autowired注解，当使用无参数构造函数或无参数静态工厂方法实例化 Bean时，Spring容器会调用这些 setter方法，以便注入 Bean的依赖项，下面给出了一个基于 setter方法注入的例子：

```
@Component
public class SetterBasedInjection {
    private final Object object;

    @Autowired
    public Object setObject(Object object) {
        this.object = object;
    }

    public Object getObject() {
        return object;
    }
}
```

基于字段注入

基于字段注入，就是在字段上加 @Autowired注解，在构造 bean之后，调用任何配置方法之前，Spring容器会立即注入这些字段，以下是一个基于字段注入的例子：

```
@Component
public class FiledBasedInjection {
    @Autowired
    private final Object object;
}
```

从上面 3种注入方式，我们可以看出：基于字段注入是最简洁的注入方式，因为它避免了添加繁冗的 getter和 setter样板代码，并且无需为类声明构造函数。但是正如 idea编译器警告的一样，基于字段注入方式势必存在某些缺点，以至于它的创造者 Spring官方不推荐使用它，下文就给出几个基于字段注入可能的缺点。

基于字段的依赖注入的缺点

容易引发NPE

因为Spring IOC容器在使用字段依赖注入时，并不会对依赖的bean是否为null做判断，因此在下面的代码中，通过 @Autowired 注入的user对象可能为空，而JVM 虚拟机在编译时也无法检测出 user为 null，只有在运行时调用user的方法时，发现 user为 null，出现空指针异常(NPE)。

```
@Component
public class FiledBasedInjection {
    private String name;
    @Autowired
    private final User user;

    public FiledBasedInjection(){
        this.name = user.getName(); // NPE
```

```
    }
}
```

缓解单一职责原则的违反

使用基于字段的注解，我们无需类之间的依赖关系，完全依赖于Spring IOC容器的管理，但是使用”基于构造器注入的方式”，我们需要手动在类代码中去编写需要依赖的类，当依赖的类越来越多，我们就能发现 code smell，这个时候就能显示的提醒我们，代码的质量是否有问题。

因此，尽管字段注入不直接负责打破单一责任原则，但它通过隐藏了和构造器注入一样发现 code smell的机会，示例代码如下：

```
@Component
public class ConstructorBasedInjection {
    private final Object object;
    private final Object object2;
            ...
    private final Object objectX;

    // @Autowired注解可以省去
    @Autowired
    public ConstructorBasedInjection(Object object,
                                     Object object2,
                                        ...       ,
                                     Object objectX) {
        this.object = object;
        this.object2 = object2;
              ...
        this.objectX = objectX;
    }
}
```

Spring官方推荐的注入方式
==========

Spring官方推荐使用基于构造器注入的方式。

另外，在国内 dubbo，RocketMQ等很多开源框架的源码都已经转向了基于构造器的注入方式，所以开发中我们应该尊重 Spring官方的推荐，尽管其他的方式可以解决，但是不推荐。

不过基于构造器注入有一个潜在的问题就是循环依赖，如下代码，ClassA初始化时依赖 ClassB，因此需要去初始化ClassB。ClassB初始化时又需要依赖 ClassA，进而转向初始化ClassA。所以就形成了经典的”循坏依赖问题”，如下代码：

```
@Component
public class ClassA {
    private final ClassB classB;

    @Lazy
    public ClassA(ClassB classB){
        this.classB = classB;
    }
}

@Component
public class ClassB {
    private final ClassA classA;

    public ClassB(ClassA classA){
        this.classA = classA;
    }
}
```

基于构造器的循环依赖
==========

对于给予构造器注入的方式，依然存在循环依赖的问题，这里主要有两种解决方案：

重构代码

既然循环依赖是代码编写带来的，最彻底的方案是重新设计结构，消除循环依赖，但是，重构代码的范围可能不可控，因此，对于测试等存在一定的回归成本，这是一种代价稍微大点的方案。

另外，代码出现循环依赖，在一定意义上（不是绝对哦）预示了 code smell：为什么会存在循环依赖？代码抽象是否合理？代码设计是否违背了 SOLID 原则？

使用 @Lazy

@Lazy 是 spring 3.0 提供的一个注解，用来表示是否要延迟初始化 bean，实现示例如下图：
![image.png](https://p6-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/02ec40e3636f4052a1b1425cd572b4aa~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1080&h=504&s=84923&e=png&b=151617)

关于 Spring如何解决基于构造器注入引发的循环依赖，欢迎阅读我的另外一篇文章：[Spring如何解决构造器注入的循环依赖?](http://cxyroad.com/”https://mp.weixin.qq.com/s?__biz=MzIwNDAyOTI2Nw==&mid=2247498018&idx=1&sn=b7dfc6aeda2bfe40d4e88808bdb93978&chksm=96c4d51ea1b35c08d7cfbe5c756028336365ab00de61d9a21e34e8ea73cf6e9a1a4cef16f1cb&token=2033710574&lang=zh_CN#rd”)

总结
==

Spring注入的方式有：基于字段注入，基于setter方法注入，基于构造器注入3种方式。

Spring官方不推荐 @Autowire使用在基于字段注入方式，推荐基于构造器注入，主要原因是：字段依赖注入容易引发 NPE 空指针异常，而构造器注入时会进行校验，如果依赖的 bean找不到就会抛出 NoSuchBeanDefinitionException，具体代码可以参考源码 `org.Springframework.beans.factory.support.ConstructorResolver#resolveAutowiredArgument`实现。

尽管 @Autowired字段注入更简单，而且还可以使用，但是它最大的问题是NPE无法在编译时期发现，因此不推荐使用，而构造器注入因其在单元测试和

不可变性方面的优势，被许多开发者视为最佳实践。

学习交流
====

文章总结不易，感谢帮忙点赞，收藏，或者公众号：猿java，持续为你呈现更多干货！

原文链接: https://juejin.cn/post/7386860512082706444