Spring 中如何控制 Bean 的加载顺序?

如果你脱口而出说添加 @Order 注解或者是实现 Ordered 接口，那么恭喜，你掉坑了。

一 @Order 注解和 Ordered 接口

在 Spring 框架中，`@Order` 是一个非常实用的元注解，它位于 `spring-core` 包下，主要用于控制某些特定上下文中组件的执行顺序或排序，但它并不直接控制 Bean 的初始化顺序。

1.1 用途

@Order 注解或者是 Ordered 接口，常见的用途主要是两种：

* **定义执行顺序**：当多个组件（如拦截器、过滤器、监听器等）需要按照特定的顺序执行时，`@Order` 注解可以用来指定这些组件的执行优先级。数值越小，优先级越高，相应的组件会更早被执行或被放置在集合的前面（`@Order` 注解接受一个整数值，这个值可以是负数、零或正数。Spring 框架提供了 `Ordered.HIGHEST_PRECEDENCE`（默认最低优先级）和 `Ordered.LOWEST_PRECEDENCE`（默认最高优先级）常量，分别对应于 Integer.MIN_VALUE和 Integer.MAX_VALUE，可以方便地设定优先级。）。
* **集合排序**：当相同类型的组件被自动装配到一个集合中时，`@Order` 注解会影响它们在这个集合中的排列顺序。

1.2 使用场景

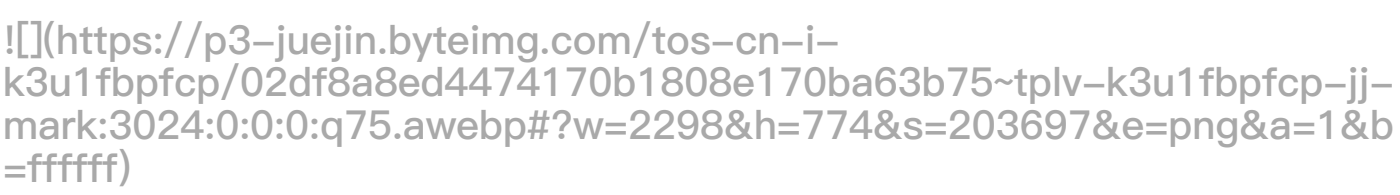
经典使用场景。

拦截器排序

在 Spring MVC 中，可以使用 `@Order` 来控制拦截器的执行顺序。

Spring Security Filters（过滤器）

在 Spring Security 中，过滤器链的顺序通过 `@Order` 来定义，确保正确的安全处理流程。这个在松哥最近的教程中和大家详细介绍过了。



Event Listeners（事件监听器）

当有多个监听同一事件的监听器时，可以通过 `@Order` 来控制它们的触发顺序。

Bean 的集合注入

当一个 Bean 依赖于一个特定类型的 Bean 集合时，带有 `@Order` 注解的 Bean 将按照指定顺序被注入。

可以看到，@Order 注解的使用场景中，主要是相同类型的 Bean 存在多个时，这多个 Bean 的执行顺序可以通过 @Order 注解或者实现 Ordered 接口来确定。

但是！！！

@Order 注解不控制初始化和加载：`@Order` 注解不直接影响 Bean 的创建和初始化过程，这些由 Spring IoC 容器基于依赖关系和配置来决定。Spring IoC 容器根据依赖关系图来决定 Bean 的初始化顺序，而不是依据 `@Order` 注解。依赖关系决定了哪些 Bean 需要在其他 Bean 初始化之前被创建。

二 如何设置 Bean 的加载顺序?

有两种方式来设置 Bean 的加载顺序。

2.1 @DependsOn

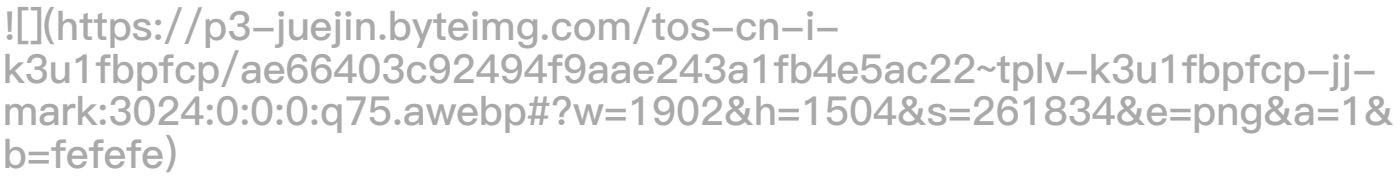
`@DependsOn` 是 Spring 框架提供的一个注解，用于指示 Spring 容器在初始化一个 Bean 之前，必须先初始化其依赖的其他 Bean。这个注解可以帮助解决 Bean 间的依赖关系，确保依赖的 Bean 已经准备就绪。

`@DependsOn` 可以放在任何一个 Spring 管理的 Bean 定义上，包括但不限于配置类、服务类、数据访问对象等。其语法如下：

```
@DependsOn({”beanName1”, ”beanName2”, ...})
public class MyBean {
    // ...
}
```

在这个例子中，`MyBean` 类声明了它依赖于名为 `beanName1` 和 `beanName2` 的 Bean。这意味着，当 Spring 容器创建 `MyBean` 的实例时，它会首先确保 `beanName1` 和 `beanName2` 已经被正确初始化。

相关的源码在 AbstractBeanFactory#doGetBean 方法中：



在创建的 Bean 的时候，先检查该 Bean 是否依赖了其他 Bean，如果依赖了，则先把其他 Bean 创建出来，然后再继续创建当前 Bean，这样就确保了 Bean 的加载顺序。

> 如果小伙伴们对这块的完整流程感兴趣，可以看松哥录制的 [Spring源码分析](http://cxyroad.com/ ”https://mp.weixin.qq.com/s/9-DQdfN8GQ2cvGwy-T3DwA”)。

2.2 BeanFactoryPostProcessor

第二种方式就是利用 BeanFactoryPostProcessor，BeanFactoryPostProcessor 的执行时机比较早，从下面这张流程图中可以看到，BeanFactoryPostProcessor 在正常的 Bean 初始化之前就执行了。

那么对于想要提前初始化的 Bean，我们可以在 BeanFactoryPostProcessor 中手动调用，类似下面这样：

```
@Component
public class MyBeanFactoryPostProcessor implements BeanFactoryPostProcessor {
    @Override
    public void postProcessBeanFactory(ConfigurableListableBeanFactory beanFactory) throws BeansException {
        //想要提前初始化的 Bean 在这里执行
        beanFactory.getBean("serviceB");
    }
}
```

三 小结

多个相同类型的 Bean 该如何确保其执行顺序？这个靠 @Order 注解或者 Ordered 接口来解决。但是这两者并不能解决 Bean 的加载顺序。Bean 的加载顺序有两种方式可以调整：

1. 通过 @DependsOn 注解加载。
2. 手动在 BeanFactoryPostProcessor#postProcessBeanFactory 方法中提

前调用 getBean 方法去初始化 Bean。

如果小伙伴们想要彻底掌握 Spring 源码，那么不妨看看松哥录制的 Spring 源码视频教程。

原文链接: https://juejin.cn/post/7375351908897554451