spring学习笔记—类路径扫描和组件管理

|  |  |
| --- | --- |
|  | 🢂 内容概览 |
|  | Why：此文档用来做什么？它存在的意义是什么？为解决什么问题？   |  | | --- | |  |   What：当前包含了那些内容？   |  | | --- | |  |   How：此文档应如何参考？   |  | | --- | |  |   Who：此文档适用于那些人员阅读参考？   |  | | --- | |  | |

目录

[1 类路径扫描&组件管理 3](#_Toc478750260)

[1.1 @Component 3](#_Toc478750261)

[1.2 配置包扫描 3](#_Toc478750262)

[1.3 过滤包扫描 3](#_Toc478750263)

[1.3.1 示例 4](#_Toc478750264)

[1.4 @Bean：在组件内定义Bean（不理解） 4](#_Toc478750265)

[1.5 定制bean名称生成规则 4](#_Toc478750266)

[1.6 定制scope 5](#_Toc478750267)

[2 使用JSR-330标准注解 5](#_Toc478750268)

[2.1 引入jar包 5](#_Toc478750269)

[2.2 @Inject 6](#_Toc478750270)

[2.3 @Named 6](#_Toc478750271)

[3 基于Java的容器化配置 7](#_Toc478750272)

[3.1 基本概念 7](#_Toc478750273)

[3.2 使用AnnotationConfigApplicationContext初始化spring容器 8](#_Toc478750274)

[3.3 使用AnnotationConfigWebApplicationConext 8](#_Toc478750275)

[3.4 使用@Bean 9](#_Toc478750276)

# 类路径扫描&组件管理

使用spring的传统方式是将基础bean配置到xml中，然后通过注解配置具体的业务bean；Spring还提供了基于JavaConfig的配置方式；

## @Component

spring提供了一系列注解用来标记需要被Spring管理的bean；@Component作为基本的stereotype注解;@Repository，@Service，@Controller作为@Component的具体原型，分别用于持久层、业务层、表现层等不同业务场景的bean；

## 配置包扫描

能够自动探测类路径下包中需要容器管理的bean；由AutowiredAnnotationBeanPostProcessor和CommonAnnotationBeanPostProcessor管理；如果需要禁用包扫描，则设置annotation-config为false;

1. 通过JavaConfig方式配置；

|  |
| --- |
|  |

1. 通过xml方式配置；

|  |
| --- |
|  |

## 过滤包扫描

通过include-filter和exclude-filter能够过滤不需要扫描的bean；过滤器作为包扫描配置的子属性，包括type和expression属性；type可取值如下：

|  |  |  |
| --- | --- | --- |
| 类型 | 描述 | 示例 |
| annotation | 默认，用来过滤注解 | org.example.XXXAnnotation |
| assignable | 过滤类 | org.example.XXXClass |
| aspectj | Aspectj类型表达式 | org.example..\*Service+ |
| regex | 用于匹配类名的正则表达式 | org\.example\.Default.\* |
| custom | 自定义过滤器，需要实现org.springframework.core.type.filter.TypeFilter接口 | org.example.MyTypeFilter |

### 示例

基于JavaConfig的配置：

|  |
| --- |
|  |

基于xml的配置：

|  |
| --- |
|  |

## @Bean：在组件内定义Bean

|  |
| --- |
|  |

## 定制bean名称生成规则

1. 默认情况下，bean的命名规则依照驼峰命名法，将类名转换为bean的名称；
2. 也可以显式指定bean的名称；

|  |
| --- |
|  |

1. 也可以通过实现BeanNameGenerator对bean的名字生成策略进行定制;

|  |
| --- |
|  |

## 定制scope

1. 实现ScopeMetaDataResolver接口并确保有一个无参构造器；
2. 在配置文件中指定解析器；

|  |
| --- |
|  |

1. 对于非singleton的bean，可能需要用到动态代理，通过scopedProxy指定动态代理类型；可以取值：no,interface,targetClass；

|  |
| --- |
|  |

# 使用JSR-330标准注解

## 引入jar包

|  |
| --- |
|  |

## @Inject

功能类似@autowire注解；

替换@autowire注解

|  |
| --- |
|  |

懒加载

|  |
| --- |
|  |

筛选Bean名称

|  |
| --- |
|  |

## @Named

功能等同于@Component注解；

|  |
| --- |
|  |

不显式指定BeanName

|  |
| --- |
|  |

# 基于Java的容器化配置

## 基本概念

1. @Bean：与xml文件中的<bean>扮演相同的角色；用于实例化、配置和初始化一个需要被IOC容器托管的对象；通常@Bean和@Configuration搭配使用；
2. @Configuration：标识一个类的基本目的是为了定义Bean；@Configuration同时允许其内部配置的bean相互调用；

基本使用示例：

|  |
| --- |
|  |

等价于xml的如下配置：

|  |
| --- |
|  |

建议：@Bean可以脱离@Configuration单独使用，此时，类内部定义的Bean不可以依赖；故建议在@C

## 使用AnnotationConfigApplicationContext初始化spring容器

1. AnnotationConfigApplicationContext可以使用@Configuration作为输入，为@Configuration以及其包含的@Bean生成Bean定义；

|  |
| --- |
|  |

1. 可以使用@Component和JSR-330定义的Bean作为输入；

|  |
| --- |
|  |

1. 通过register方法注册配置；

|  |
| --- |
|  |

1. 通过scan方法自动扫描配置；

|  |
| --- |
|  |

|  |
| --- |
|  |

等价于：

|  |
| --- |
|  |

## 使用AnnotationConfigWebApplicationConext

AnnotationConfigWebApplicationContext是WebApplicationContext的变体；能够通过ContextLoaderListener或者DispatchServlet进行配置，配置示例如下：

|  |
| --- |
|  |

## 使用@Bean