原 SpringBoot-向容器注册Bean的多种方式

2018年06月15日 00:15:15 东京易冷 阅读数 5769 文章标签： SpringBoot 更多

摘要

向Spring容器注册Bean有多种方式，本文介绍下面的几种。

- @ComponentScan
- @Bean
- @Import

通过@ComponentScan注册Bean

Spring容器会扫描@ComponentScan配置的包路径，找到标记@Component注解的类加入到Spring容器。

效果等同于XML配置文件中的 `<context:component-scan base-package="包名">`

常用属性名	类型	说明
includeFilters	Filter[]	指定扫描导入类型的过滤规则
excludeFilters	Filter[]	指定扫描排除类型的过滤规则

java8之后一个类可以标记多个@ComponentScan扫描规则

@Component说明

常见继承：
- @Configuration：标记类为配置类，常与@ComponentScan或@Bean注解一起使用
- @Controller
- @Repository
- @Service

通过@Bean注册Bean

标记在方法上，将方法返回值注册到Spring容器，类型为返回值类型，id默认为方法名。

效果等同于XML配置文件中的 `<bean id="beanName" class="className"/>`

通过@Import注册Bean

- 直接注册指定类

```
1  // 启动类
2  @Import({ ImportTest.class })
3  public class RegistryBean {
4
5      public static void main(String[] args) throws Exception {
6          ConfigurableApplicationContext context = SpringApplication.run(RegistryBean.class, args);
7          String[] beanNames = context.getBeanDefinitionNames();
8          for (String beanName : beanNames) {
9              System.out.println(beanName);
```

```
        }
    }

}

public class ImportTest {
}
```

- 配合ImportSelector接口注册指定类

```
public class ImportSelectorTest implements ImportSelector {

    // ImportSelector接口定义的方法
    // Spring容器会传入当前标记@Import的类的全部注解元数据，用于读取注解中的配置
    // 返回需要注册的全类名，Spring容器会自动注册这些类
    // 注意：不能返回null
    @Override
    public String[] selectImports(AnnotationMetadata importingClassMetadata) {
        // 返回想要注册的全类名
        return new String[] { ImportBeanTest.class.getName() };
    }

}

public class ImportBeanTest {
}
```

同时修改启动类上注解 `@Import({ ImportTest.class })` 为 `@Import({ ImportTest.class, ImportSelectorTest .class })`

- 配合ImportBeanDefinitionRegisterar接口注册指定类

```
public class ImportBeanDefinitionRegistrarTest implements ImportBeanDefinitionRegistrar {

    // ImportBeanDefinitionRegistrar接口定义的方法
    // Spring容器会传入当前标记@Import的类的全部注解元数据和Bean定义信息注册类
    // Spring容器会按照注册类中注册的Bean定义信息进行实例化Bean组件
    @Override
    public void registerBeanDefinitions(AnnotationMetadata importingClassMetadata, BeanDefinitionRegistry registry) {
        // 包装Bean定义信息并进行注册
        RootBeanDefinition beanDefinition = new RootBeanDefinition(String.class);
        registry.registerBeanDefinition("ImportBeanDefinitionRegistrar测试类", beanDefinition);
    }

}
```

同时修改启动类上注解 `@Import({ ImportTest.class })` 为 `@Import({ ImportTest.class,ImportSelectorTest.class, ImportBeanDefinitionRegistrarTest.class })`

- 输出结果

执行启动类

```
org.springframework.context.annotation.internalConfigurationAnnotationProcessor
org.springframework.context.annotation.internalAutowiredAnnotationProcessor
org.springframework.context.annotation.internalRequiredAnnotationProcessor
org.springframework.context.annotation.internalCommonAnnotationProcessor
org.springframework.context.event.internalEventListenerProcessor
org.springframework.context.event.internalEventListenerFactory
registryBean
org.springframework.boot.autoconfigure.internalCachingMetadataReaderFactory
com.ly.bean.ImportTest
com.ly.bean.ImportBeanTest
ImportBeanDefinitionRegistrar测试类
```

输出结果就是Spring容器全部bean的名字，最下面三行就是通过以上三种方式注册的bean组件