

使用Java方式配置Spring

完全不使用Spring的xml配置，全权交给Java来做。

JavaConfig是Spring的子项目，在Spring4之后，成为了一个核心功能。

- 实体类

```
1 package com.kuang.pojo;
2
3 import org.springframework.beans.factory.annotation.Value;
4
5 import java.io.Serializable;
6
7 public class User implements Serializable {
8     @Value("Geekst")
9     private String name;
10
11     public String getName() {
12         return name;
13     }
14
15     public void setName(String name) {
16         this.name = name;
17     }
18
19     @Override
20     public String toString() {
21         return "User{" +
22             "name='" + name + '\'' +
23             '}';
24     }
25 }
```

- 编写配置类

```
1 package com.kuang.config;
2
3 import com.kuang.pojo.User;
4 import org.springframework.context.annotation.Bean;
5 import org.springframework.context.annotation.ComponentScan;
6 import org.springframework.context.annotation.Configuration;
7 import org.springframework.context.annotation.Import;
8
9 // @Configuration 代表这是一个配置类，和之前的xml一样
10 // @ComponentScan 扫描包
11 // @Import 导入配置类
12 // 这个会被Spring容器托管，注册到容器中，因为它本身也是一个@Component
13 @Configuration
14 @ComponentScan("com.kuang.pojo")
15 @Import(MyConfig2.class)
16 public class MyConfig {
17 }
```

```

18 //注册一个bean，相当于我们之前写的一个bean标签
19 //id: 这个方法的名字
20 //class: 方法的返回值
21 @Bean
22 public User getUser() {
23     return new User(); //就是返回要注入到bean的对象
24 }
25 }

```

```

1 @Configuration
2 public class MyConfig2 {
3 }

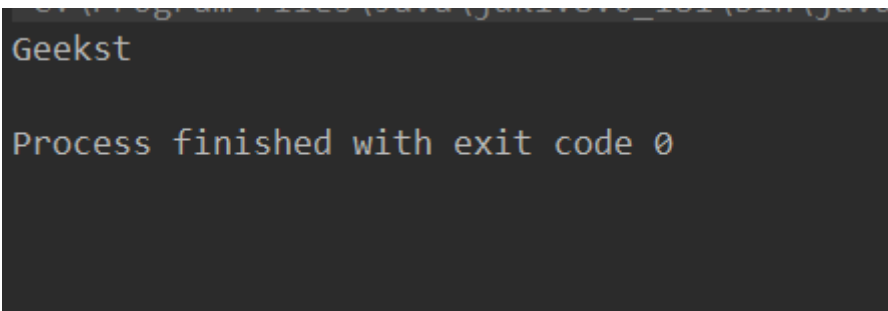
```

- 运行测试

```

1 @Test
2 public void test(){
3     ApplicationContext context = new
AnnotationConfigApplicationContext(MyConfig.class);
4     User user = context.getBean("getUser", User.class);
5     System.out.println(user.getName());
6
7 }

```



```

Geekst
Process finished with exit code 0

```

这种纯Java的配置方式，在Spring Boot中随处可见，所以十分重要！