

IOC创建对象的方式

1 默认使用无参构造

- User实体类

```
1 package com.kuang.pojo;
2
3 import java.io.Serializable;
4
5 public class User implements Serializable {
6     private String name;
7
8     public User() {
9         system.out.println("User的无参构造");
10    }
11
12    public String getName() {
13        return name;
14    }
15
16    public void setName(String name) {
17        this.name = name;
18    }
19
20    public void show() {
21        system.out.println("name = " + name);
22    }
23 }
```

这里我们给无参构造写一个输出，方便后续验证。

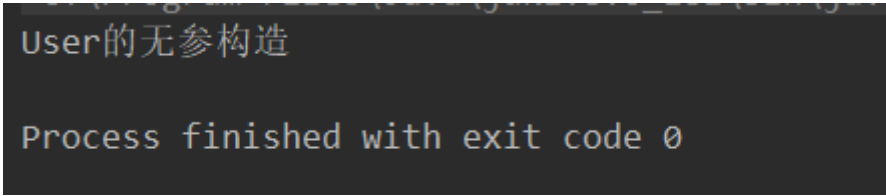
- xml文件中注册

```
1 <bean id="user" class="com.kuang.pojo.User">
2     <property name="name" value="Spring"/>
3 </bean>
```

- 运行测试

①当使用new对象的时候：

```
1 User user = new User();
```



User的无参构造

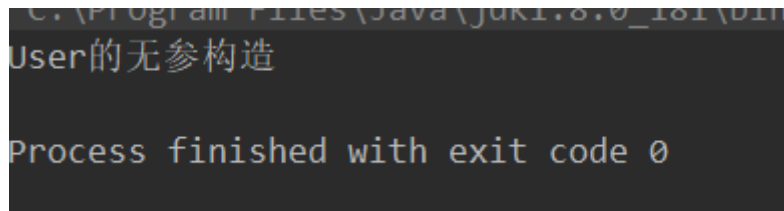
Process finished with exit code 0

②当使用Spring管理的时候

```

1 ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
2
3 User user = (User) context.getBean("user");

```



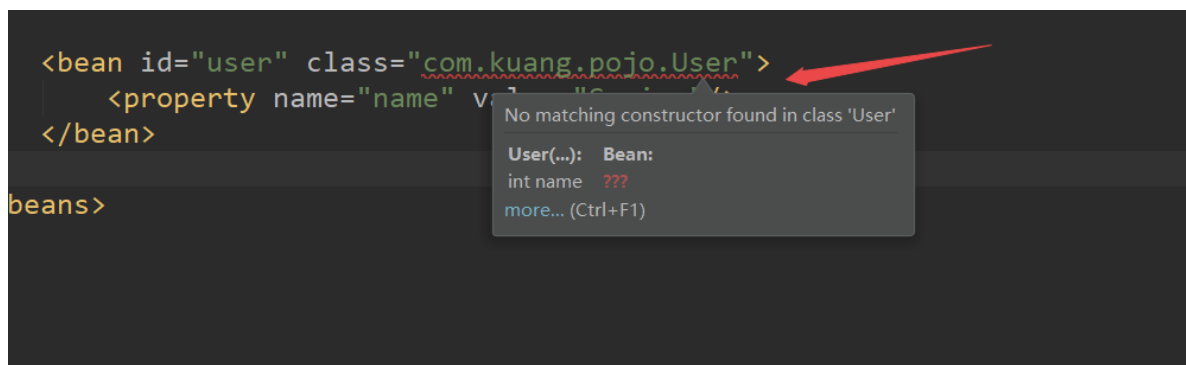
C:\Program Files\Java\jdk1.8.0_181\bin
 User的无参构造
 Process finished with exit code 0

- 总结
 - 对比可以看出，Spring在创建对象的时候默认使用无参构造。
 - 细心的人可以发现，注释掉下面的 `User user = (User) context.getBean("user");` 后，也会打印出如上的结果。这是因为在扫描文件后，Spring就将所有的bean注册完成放进容器中了，我们的“user”只是调用后实例的对象。
 - 如果用 `User user2 = (User) context.getBean("user");`
 可以发现，`user == user2`，因为始终容器中只有一个User。（默认单例模式，后面作用域会说到）
- 当我们给无参构造“抹去的时候”，就会报错，进一步说明了默认是无参构造

```

1 public User(String name) {
2     System.out.println("User的无参构造");
3 }

```



`<bean id="user" class="com.kuang.pojo.User">`
`<property name="name" value="Spring"/>`
`</bean>`
 beans>

No matching constructor found in class 'User'
 User(...): Bean:
 int name ???
 more... (Ctrl+F1)

2 使用有参构造创建对象

2.1 下标

- 修改一下User类:

```

1 public User(String name) {
2     this.name = name;
3     System.out.println("name = " + name );
4 }

```

- xml文件

```

1 <bean id="user" class="com.kuang.pojo.User">
2     <constructor-arg index="0" value="Spring"/>
3 </bean>

```

constructor-arg: 代表使用指定的有参构造

index: 指的是构造函数中第几个参数。

value: 给这个参数赋初始值

- 测试运行

```
name = Spring
Process finished with exit code 0
```

2.2 类型 (不建议)

只需要在xml中配置。

```
1 <bean id="user" class="com.kuang.pojo.User">
2     <constructor-arg type="java.lang.String" value="SpringByType"/>
3 </bean>
```

运行测试:

```
name = SpringByType
Process finished with exit code 0
```

- type: 参数的Java类型
- value: 给参数赋值

但是, 会出现一个问题: 当两个参数都是同一类型的时候, 就无法识别了。

2.3 参数名

改动xml文件:

```
1 <bean id="user" class="com.kuang.pojo.User">
2     <constructor-arg name="name" value="SpringByName"/>
3 </bean>
```

- name: 参数名
- value: 给参数赋值

测试运行:

```
name = SpringByName
Process finished with exit code 0
```

3 总结

在配置文件加载的时候，容器中管理的对象就已经初始化了。