

- 3.2、基于XML管理bean
 - 3.2.1、搭建子模块spring6-ioc-xml
 - 3.2.2、实验一：获取bean
 - 3.2.3、实验二：依赖注入之setter注入
 - 3.2.4、实验三：依赖注入之构造器注入
 - 3.2.5、实验四：特殊值处理
 - 3.2.6、实验五：为对象类型属性赋值
 - 3.2.7、实验六：为数组类型属性赋值
 - 3.2.8、实验七：为集合类型属性赋值
 - 3.2.9、实验八：p命名空间
 - 3.2.10、实验九：引入外部属性文件
 - 3.2.11、实验十：bean的作用域
 - 3.2.12、实验十一：bean生命周期
 - 3.2.13、实验十二：FactoryBean
 - 3.2.14、实验十三：基于xml自动装配
- 3.3、基于注解管理bean (☆)

- 1、把 spring-first 的 pom.xml 依赖的注入放到父工程；
创建子模块 spring6-ioc-xml；
创建 user 类、添加 bean.xml 配置文件、复制 log4j2 的配置文件

```

1 package com.atguigu.spring6.iocxml;
2
3 import org.apache.logging.log4j.core.util.JsonUtils;
4
5 public class User {
6
7     private String name;
8     private Integer age;
9
10    public void run() {
11        System.out.println("run.....");
12    }
13 }
  
```

- 2、获取 bean 的三种方式
 - 1) 通过 id 获取
- bean.xml 文件进行配置

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www
5
6       <!--user 对象创建-->
7       <bean id="user" class="com.atguigu.spring6.iocxml.User"></bean>
8 </beans>
  
```

创建测试类

```
spring6 spring6-ioc-xml src main java com atguigu spring6 iocxml TestUser main
pom.xml (spring6-ioc-xml) bean.xml TestUser.java User.java
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class TestUser {
7
8     public static void main(String[] args) {
9         ApplicationContext context = new
10             ClassPathXmlApplicationContext( configLocation: "bean.xml");
11         // 根据id获取bean
12         User user = (User)context.getBean( name: "user");
13         System.out.println("根据id获取bean:"+user);
14     }
15 }
```

2) 根据类型获取 bean

```
spring6 spring6-ioc-xml src main java com atguigu spring6 iocxml TestUser main
pom.xml (spring6-ioc-xml) bean.xml TestUser.java log4j2.xml User.java
15 //2 根据类型获取bean
16 User user2 = context.getBean(User.class);
17 System.out.println("2 根据类型获取bean: "+user2);
18
19 }
20
Run: TestUser
2022-12-12 09:01:29 751 [main] DEBUG org.springframework.context.support.ClassPa
2022-12-12 09:01:29 904 [main] DEBUG org.springframework.beans.factory.xml.XmlBe
2022-12-12 09:01:29 949 [main] DEBUG org.springframework.beans.factory.support.D
1 根据id获取bean: com.atguigu.spring6.iocxml.User@2374d36a
2 根据类型获取bean: com.atguigu.spring6.iocxml.User@2374d36a
```

3) 根据类型+id 获取

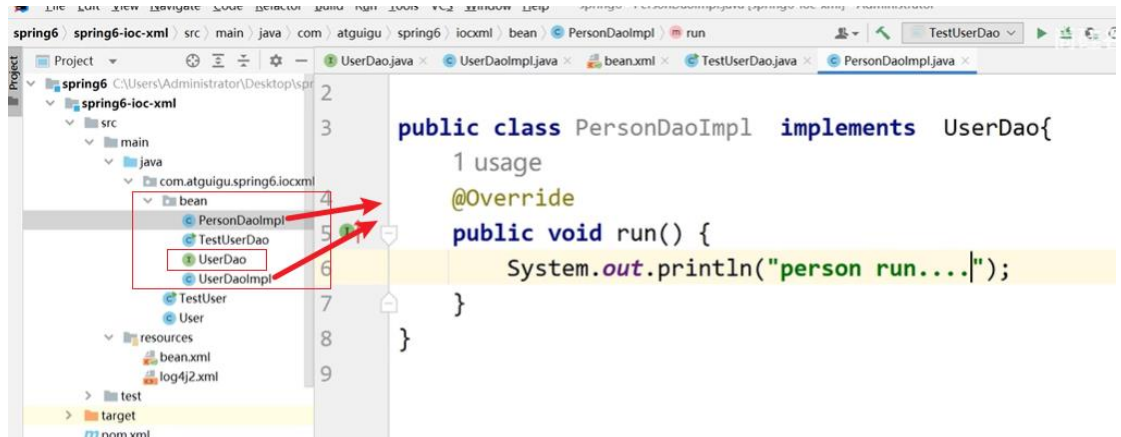
```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help spring6 - TestUser.java [spring6-ioc-xml] - Administrator
spring6 spring6-ioc-xml src main java com atguigu spring6 iocxml TestUser main
pom.xml (spring6-ioc-xml) bean.xml TestUser.java log4j2.xml User.java
10 ClassPathXmlApplicationContext( configLocation: "bean.xml");
11 //1 根据id获取bean
12 User user1 = (User)context.getBean( name: "user");
13 System.out.println("1 根据id获取bean: "+user1);
14
15 //2 根据类型获取bean
16 User user2 = context.getBean(User.class);
17 System.out.println("2 根据类型获取bean: "+user2);
18
19 //3 根据id和类型获取bean
20 User user3 = context.getBean( name: "user", User.class);
21 System.out.println("3 根据id和类型获取bean: "+user3);
22 }
23
24
```

注意根据类型获取 bean 的话，class 必须唯一：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
5
6       <!--user对象创建-->
7       <bean id="user" class="com.atguigu.spring6.iocxml.User"></bean>
8
9       <bean id="user1" class="com.atguigu.spring6.iocxml.User"></bean>
10 </beans>
```

```
Exception in thread "main" org.springframework.beans.
factory.NoUniqueBeanDefinitionException:
No qualifying bean of
type 'com.atguigu.spring6.iocxml.User'
available: expected single matching bean but found 2: user,user1
```

根据类型获取失败情况二：一个接口有多个实现类



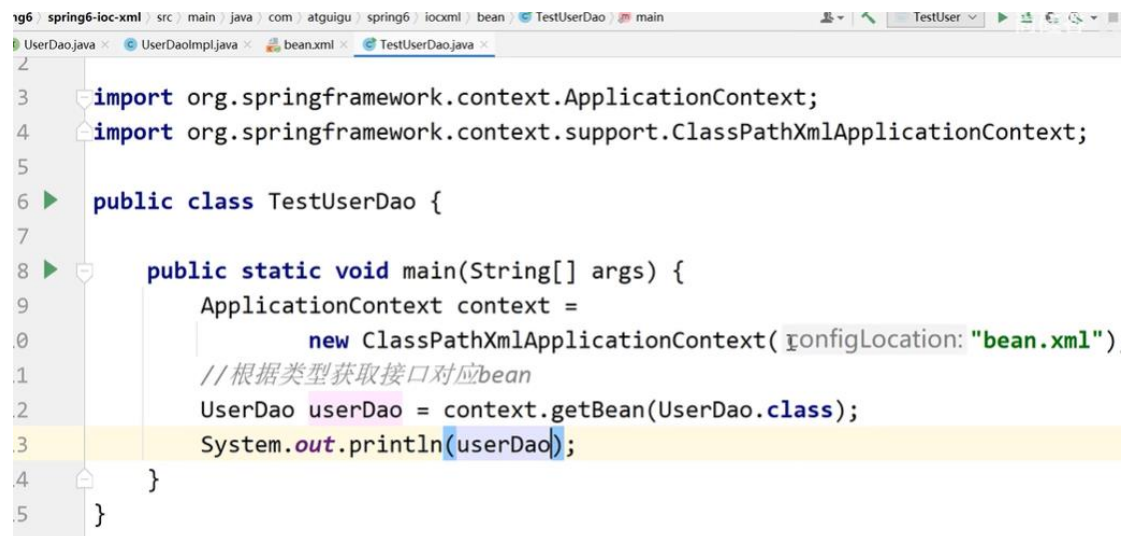
```
2 public class PersonDaoImpl implements UserDao{
3     1 usage
4     @Override
5     public void run() {
6         System.out.println("person run...|");
7     }
8 }
9
```

```
<!--1 获取bean演示, user对象创建-->
<bean id="user" class="com.atguigu.spring6.iocxml.User"></bean>

<bean id="user1" class="com.atguigu.spring6.iocxml.User"></bean>

<!--2 一个接口实现类获取过程-->
<bean id="userDaoImpl" class="com.atguigu.spring6.iocxml.bean.UserDaoImpl">
</bean>
<bean id="personDaoImpl" class="com.atguigu.spring6.iocxml.bean.PersonDaoImpl"
</bean>
```

运行：



```

1  import org.springframework.context.ApplicationContext;
2  import org.springframework.context.support.ClassPathXmlApplicationContext;
3
4  public class TestUserDao {
5
6      public static void main(String[] args) {
7          ApplicationContext context =
8              new ClassPathXmlApplicationContext("bean.xml")
9              // 根据类型获取接口对应bean
10             UserDao userDao = context.getBean(UserDao.class);
11             System.out.println(userDao);
12         }
13     }

```

结论

根据类型来获取bean时，在满足bean唯一性的前提下，其实只是看：『对象 instanceof 指定的类型』的返回结果，只要返回的是true就可以认定为和类型匹配，能够获取到。

java中，instanceof运算符用于判断前面的对象是否是后面的类，或其子类、实现类的实例。如果是返回true，否则返回false。也就是说：用instanceof关键字做判断时，instanceof 操作符的左右操作必须有继承或实现关系