5，spring注入对象类型属性（重点）

（1）创建service类和dao类

（（1））在service得到dao对象

（2）具体实现过程

（（1））在service里面把dao作为类型属性

userservice类：

```
package cn.java.service;
public class userservice {
	// 定义一个userdao类型的属性
	private userdao userdao;
	// 生成userdao的set和get方法
	public userdao getUserdao() {
		return userdao;
	}
	public void setUserdao(userdao userdao) {
		this.userdao = userdao;
	}
	public void add() {
		System.out.println("service.....");
		// 在service里面得到dao类对象，才能调用dao里面的方法
		userdao.add();
	}
}
```

userdao类：

```
package cn.java.service;
public class userdao {
	public void add(){
		System.out.println("dao.....");
	}
}
```

（（2））生成dao类型属性的set方法

测试代码类：

```
package cn.java.service;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class test {
@Test
public void test1(){
	ApplicationContext context =
```

```
                new ClassPathXmlApplicationContext("applicationcontext.xml");
        userservice sUserservice = (userservice)context.getBean("userservice");
        sUserservice.add();
}
}
```

（（3））在配置文件中写入配置信息

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
        <!-- 注入对象类型属性 -->
        <!-- 配置service和dao对象 -->
        <bean id="userdao" class="cn.java.service.userdao"></bean>
        <bean id="userservice" class="cn.java.service.userservice">
                <!-- 注入dao对象； name属性值：service类里面属性名称； 现在不要写value
属性，因为刚才是一个字符串，而现在是一个对象；
                        而是要写ref属性;其中ref中dao配置bean标签中的id值 -->
                <property name="userdao" ref="userdao"></property>
        </bean>
</beans>
```

6，p名称空间注入

（1）引入p名称空间（在applicationcontext.xml配置文件中）

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

（2）使用p名称空间

①普通属性：（p：属性名称=“属性值”）

②对象类型属性：（p：属性名称-ref=“已经使用bean声明的实体对象的id值”）

（3）具体代码

①person实体类：

```
package cn.java.persons;
public class person {
private String pname;
public String getPname() {
        return pname;
}
public void setPname(String pname) {
        this.pname = pname;
}
public void test1(){
        System.out.println("person's name="+pname);
}
}
```

②applicationcontext.xml配置信息：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
		<!-- p名称空间注入 -->
		<bean id="person" class="cn.java.persons.person" p:pname="蔺明俊">
</bean>
</beans>
```

③具体的测试代码：

```
package cn.java.persons;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class tests {
@Test
public void test1(){
	ApplicationContext context =
			new ClassPathXmlApplicationContext("applicationcontext.xml");
	person person = (person)context.getBean("person");
	person.test1();
}
}
```

7，spring注入复杂数据（会使用）

（1）数组

（2）list集合

（3）map集合

（4）properties类型

（5）具体的代码如下：

applicationcontext.xml配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:p="http://www.springframework.org/schema/p"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
		<!-- 注入复杂类型的属性值 -->
	<!-- 配置对应实体类 -->
	<bean id="person" class="cn.java.persons.person">
		<!-- 数组类型注入 -->
```

```
        <property name="arrs">
            <list>
                <value>java</value>
                <value>python</value>
                <value>php</value>
                <value>c#</value>
            </list>
        </property>
        <!-- list类型注入 -->
        <property name="list">
            <list>
                <value>计算机</value>
                <value>水杯</value>
                <value>苹果</value>
                <value>风扇</value>
            </list>
        </property>
        <!-- map类型注入 -->
        <property name="map">
            <map>
                <entry key="a" value="anqili"></entry>
                <entry key="b" value="linmingjun"></entry>
                <entry key="c" value="liujiahui"></entry>
            </map>
        </property>
        <!-- properties类型注入 -->
        <property name="properties">
            <props>
                <prop key="driverclass">com.mysql.jdbc.Driver</prop>
                <prop key="username">root</prop>
            </props>
        </property>
    </bean>
</beans>
```

person实体类：

```
package cn.java.persons;
import java.util.List;
import java.util.Map;
import java.util.Properties;
public class person {
private String pname;
public String getPname() {
    return pname;
}
public void setPname(String pname) {
    this.pname = pname;
```

```
}
public void test1(){
	System.out.println("person's name="+pname);
}
private String []arrs;
private List<String>list;
private Map<String, String>map;
//导入util中的那个包
private Properties properties;
public String[] getArrs() {
	return arrs;
}
public void setArrs(String[] arrs) {
	this.arrs = arrs;
}
public List<String> getList() {
	return list;
}
public void setList(List<String> list) {
	this.list = list;
}
public Map<String, String> getMap() {
	return map;
}
public void setMap(Map<String, String> map) {
	this.map = map;
}
public Properties getProperties() {
	return properties;
}
public void setProperties(Properties properties) {
	this.properties = properties;
}
//输出复杂类型中的内容
public void test11(){
	System.out.println("arrs:"+arrs[0]);
	System.out.println("list:"+list);
	System.out.println("map:"+map);
	System.out.println("properties:"+properties);
}
}
```

测试代码：

```
package cn.java.persons;

import org.junit.Test;
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class tests {
@Test
public void test1(){
	ApplicationContext context =
			new ClassPathXmlApplicationContext("applicationcontext.xml");
	person person = (person)context.getBean("person");
	person.test1();
	System.out.println("============================");
	person.test11();
}
}
```