1，把对象的创建交给spring进行管理

2，ioc操作俩部分

（1）ioc的配置文件方式

（2）ioc的注解方式

一，spring的bean管理（基于xml文件的方式）

1，bean实例化三种方式

①使用类的无参构造创建

注意：这就要求在需要创建的实体类中提供没有参数的构造函数，否则会出现异常。

②使用静态工厂创建

创建静态的方法，返回类的对象。

代码：application.xml

```
<?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">
       <!-- 使用静态方法创建对象 -->
       <bean id="bean1" class="beans.bean1_factory" factory-method="getBean1">
</bean>
</beans>
```

静态工厂类：

```
package beans;

public class bean1_factory {
//静态方法返回bean1对象
       public static bean1 getBean1(){
              return new bean1();
       }
}
```

测试代码：

```
package beans;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import hello.test_hello;
public class tes {
@Test
public void testbean1(){
       //加载spring的配置文件，根据创建对象
                     ApplicationContext context = new
```

```
ClassPathXmlApplicationContext("applicationcontext.xml");
                //得到配置创建的对象
                bean1 hello = (bean1)context.getBean("bean1");
                hello.add();
}
}
```

③使用实例工厂创建

创建不是静态的方法，返回类对象。

application.xml的配置文件：

```
<?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">
      <!-- 使用实例方法创建对象 -->
      <!-- 先创建工厂对象 -->
      <bean id="bean3factory" class="beans.bean3factory"></bean>
      <bean id="bean3" factory-bean="bean3factory" factory-method="getBean3">
</bean>
</beans>
```

bean3实例：

```
package beans;
public class bean3 {
public void muty(){
      System.out.println(10*52);
}
}
```

factory工厂：

```
package beans;
public class bean3factory {
public bean3 getBean3(){
      return new bean3();
}
}
```

测试代码：

```
package beans;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class test {
@Test
public void tests(){
      ApplicationContext context=
```

```
                new ClassPathXmlApplicationContext("applicationcontext.xml");
        bean3 bean3 = (bean3)context.getBean("bean3");
        bean3.muty();
}
}
```

2，bean标签常用属性

（1）id属性：起的名称，id值理论上可以任意命名。

注意：①id属性值，不能包含中文、特殊符号（美元符、&、#等）；

②可以根据id值得到我们配置的对象。

（2）class属性：创建对象所在类的全路径。

（3）name属性：功能和id属性是一样的。

注意：name和id的区别：id属性值不能包含特殊的符号，但是在name属性值里面可以包含特殊符号。

（4）scope属性：设置bean的作用范围。其属性值有如下几个：

①singletton：默认值，单例的；

②prototype：多例的；

③request：web项目中，spring创建一个bean的对象，将对象存入到request域中；

④session：web项目中，spring创建一个bean的对象，将对象存入到session域中；

⑤globalSession：web项目中，应用在porlet环境，如果没有porlet环境nameglobalSession相当于session。

3，属性注入方式介绍

在使用spring创建对象的时候，给实体中的属性注入值。

（1）属性注入的三种方式介绍

①使用set的方式注入

```
public class User {
  private String name;
 public void setName(String name) {
   this.name = name;
 }
}
User user = new User();
user.setName("abcd");
```

②使用有参数的构造函数注入

```
public class User {
  private String name;
  public User(String name) {
    this.name = name;
  }
}
User user = new User("lucy");
```

③使用接口的方式注入

```
public interface Dao {
  public void delete(String name);
}
public class DaoImpl implements Dao {
  private String name;
  public void delete(String name) {
    this.name = name;
  }
}
```

注意：spring框架中，只是支持set和有参数构造方法注入。

4，spring注入属性（有参构造和set方法）

（1）有参数构造方式注入参数

①user实体类

```
package cn.java.entity;
public class user {
private String name;
private int id;
public user() {
	super();
}
public user(String name, int id) {
	super();
	this.name = name;
	this.id = id;
}
public String getName() {
	return name;
}
public void setName(String name) {
	this.name = name;
}
public int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
```

```
public void test1(){
	System.out.println("demo1....."+name+" "+id);
}
}
```

②application.xml配置文件

```
<?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">

	<!-- 使用有参数构造方式进行参数注入 -->
	<bean id="user" class="cn.java.entity.user">
		<!-- 使用有参数的构造方式注入参数 -->
		<constructor-arg value="安启力" index="0"></constructor-arg>
		<constructor-arg value="1611010201" index="1"></constructor-arg>
	</bean>
</beans>
```

③测试代码

```
package cn.java.entity;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import beans.bean3;
public class test_users {
@Test
public void come1(){
	ApplicationContext context=
			new ClassPathXmlApplicationContext("applicationcontext.xml");
	user bean3 = (user)context.getBean("user");
	System.out.println(bean3.getName());
	bean3.test1();
}
}
```

（2）set方式注入参数

①book实体类

```
package cn.java.book_test;
public class book {
private String booname;
//生成set、get方法
public String getBooname() {
	return booname;
}
```

```
public void setBooname(String booname) {
	this.booname = booname;
}
public book() {
	super();
}
//测试方法
public void book_testbook(){
	System.out.println("书名为："+booname);
}
}
```

②application.xml配置文件

```
<?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">
		<!-- 使用set方式进行参数的注入 -->
		<!-- 创建book对象 -->
		<bean id="book" class="cn.java.book_test.book">
		<!-- 设置对象的属性值
		name属性值是类里面定义的属性名称；
		value属性：设置具体的属性值
		-->
		<property name="booname" value="圣经"></property>
		</bean>
</beans>
```

③测试代码

```
package cn.java.book_test;
import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class book_test {
@Test
public void tets1(){
	ApplicationContext context =
			new ClassPathXmlApplicationContext("applicationcontext.xml");
	book book = (book)context.getBean("book");
	book.book_testbook();
}
}
```