自动装配bean

1 概念

- 自动装配是Spring满足bean依赖的一种方式
- Spring会在上下文之间自动寻找，并自动给bean装配属性

2 Spring装配bean的三种方式

2.1 xml配置

之前的bean标签。

2.2 Java显式配置

2.3 隐式自动装配【重点】

搭建环境：

- 一个人拥有猫和狗

```java
package com.kuang.pojo;

import java.io.Serializable;

public class Dog implements Serializable {
    public void show(){
        System.out.println("wang~");
    }
}
```

```java
package com.kuang.pojo;

import java.io.Serializable;

public class Cat implements Serializable {
    public void show(){
        System.out.println("miao~");
    }
}
```

```java
package com.kuang.pojo;

import java.io.Serializable;

public class People implements Serializable {
    private Dog dog;
    private Cat cat;
    private String name;

    /* .......getter和setter......
    ......toString()......
    */
}
```

3 byName自动装配

```xml
<bean id="people" class="com.kuang.pojo.People" autowire="byName">
    <property name="name" value="Geekst"/>
</bean>
```

```
miao~
wang~

Process finished with exit code 0
```

```
byName:会自动在容器上下文中查找，和自己对象set方法后的值对应的bean id
```

- 需要保证所有bean的id唯一

```java
public Dog getDog() {
    return dog;
}

public void setDog(Dog dog) {
    this.dog = dog;
}

public Cat getCat() {
    return cat;
}

public void setCat(Cat cat) {
    this.cat = cat;
}

public String getName() {
    return name;
}
```

```xml
<bean id="cat" class="com.kuang.pojo.Cat"/>

<bean id="dog" class="com.kuang.pojo.Dog"/>
```

4 byType自动装配

```xml
<!--
byType:会自动在容器上下文中查找，和自己对象属性类型相同的bean
-->
<bean id="people" class="com.kuang.pojo.People" autowire="byType">
    <property name="name" value="Geekst"/>
</bean>
```

甚至可以不写dog和cat的bean id：

```xml
<bean class="com.kuang.pojo.Cat"/>

<bean class="com.kuang.pojo.Dog"/>
```

- 需要保证所有的class唯一（不可以出现多个bean，配同一个class）

5 注解自动装配 @Autowired

- 导入约束：context

- 配置注解的支持：context:annotation-config/

```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"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd">

    <context:annotation-config/>

</beans>
```

- 使用
  - 直接在属性或者set方法上使用

```java
@Autowired
private Dog dog;

@Autowired
private Cat cat;
```

  - 可以忽略set方法（不写），前提是这个自动装配的属性在容器中（xml）存在，且名字符合byName的规范。
- 科普
  - @Nullable 字段如果使用了这个注解，代表这个字段可以为NULL
  - 当@Autowired自动装配的环境比较复杂（比如存在多个id，但是是用一个class），自动装配无法通过一个注解（@Atutowired）明确知道使用哪个id的bean，这时我们可以使用@Qualifier(value = "xxx") 来指定某个id的bean进行装配。

```java
@Qualifier(value = "cat11")
@Autowired
private Cat cat;
```

```xml
<bean id="cat" class="com.kuang.pojo.Cat"/>
<bean id="cat11" class="com.kuang.pojo.Cat"/>
<bean id="cat22" class="com.kuang.pojo.Cat"/>
```

6 @Resource自动装配

java的注解。

```java
    @Resource
    private Dog dog;

    @Resource
    private Cat cat;
```

- 也可以这么使用

```
@Resource(name = "cat22")
private Dog dog;
```

用于指定某个id。

- 当只存在一个唯一的class的时候，bean的名字可以随意。（但是建议不要这样）

```
<bean id="cat11" class="com.kuang.pojo.Cat"/>
<bean id="dog232" class="com.kuang.pojo.Dog"/>
```

@Autowired和@Resource的区别：

- 都是用来自动装配，都可以放在属性字段上
- @Autowired通过byType的方式，当类型大于1时再根据byName注入，而且必须要求这个对象存在【常用】
- @Resource默认通过byName的方式实现，如果找不到名字，就通过byType实现。都找不到则报错。